PERSONAL COMPUTER MAGAZINE for MZ, X1 and X68000

Oh!X

特集 急接近! SX-WINDOW

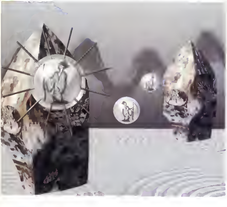
1

1991

付録5"2HDディスク
謹賀新年PRO-68K

ヒジュアルシェルの拡張 VS2 X/グラフ作成ツールMichael
半影つきレイトレHASH X/Z'sSTAFF拡張機能/カード型DB
MUSICDRV XV2.0/CARDDRV X&カードゲームKLONDIKE
SX-WINDOW開発資料/ツール/15パズル/LIFE GAME他

SOFT BANK オー・ノ・エックス 特別定価780円










Oh!X

CONTENTS



〈スタッフ〉

1991 JAN.
1

SHARP



シャープオリジナルソフトウェア X68000

ハイアビリティを実証する多彩なソフトウェア。お望みのワークベンチへ。

ドロー編集、WYSIWYG印刷、こんなC.G.ツールが欲しかった。

本格的なロゴタイプやPOPを簡単に作成できるグラフィックツールです。優先順位が任意に指定できるドローセル、ペイントセル、テキストセルの3つの仮想セルで、目的にあった自由なグラフィックが駆使できます。また印刷は、画面イメージがそのまま印刷イメージとなるWYSIWYG(What You See Is What You Get)を実現。A6 A5 A4 A3 B6 B5 B4 葉書サイズで8色カラー印字できます。



〈ドローセル〉

〈ペイントセル〉

〈テキストセル〉

CANVAS PRO-68K CZ-243GS 標準価格29,800円(税別)

■主として個人用のさまざまなジャンルのデータが収められているドローグラフィックデータ集です



CANVAS PRO-68K ドローグラフィックライブラリ VOL.1
CZ-258GS 標準価格8,800円(税別)

■主としてビジネス用のさまざまなジャンルのデータが収められているドローグラフィックデータ集です



CANVAS PRO-68K ドローグラフィックライブラリ VOL.2
CZ-256GS 標準価格8,800円(税別)

バージョンアップされたCコンパイラと、強力なBASTOCチェッカー。

ソースコードデバッガをはじめ、各種開発ツールを強化。バージョンアップされたCコンパイラ。

Cのソースレベルでデバッグできる「ソースコードデバッガ」を搭載したほか、各種開発ツールを強化した総合開発ツールです。ライブラリはHuman68k ver2.0の拡張DOSコールもサポートしているなど、よりX68000のハードウェアを生かせる豊富なライブラリ(630種以上)となっています。C言語の標準であるANSI規格準拠をさらに強化。「プログラム保守ユーティリティ(MAKE)」や「ライブラリ」など各種ツールを追加しました。その他「アセンブラ」「リンカ」「デバッガ」「ソースコードデバッガ」「コンバータ」などのツールが添付されています。



COMPILER PRO-68K CZ-245LS 標準価格44,800円(税別)

トラブルエラーの悩み解消!
「XBAStoC」の強力ツールの登場です。

X-BASICプログラムのコンパイル時、発見しづらいトラブルエラーに悩まされていたプログラムの問題点をひとつひとつ指摘。エラーとなる個所を指摘するだけでなく、注意項目も指摘します。これにより X-BASICでは実行できたのにコンパイルするとエラーが発生する、といったプログラムの修正が簡単にできます。



XBAStoC CHECKER PRO-68K
CZ-260LS 標準価格5,800円(税別)



ビジネスツール

Hyperword

CYBERNOTE PRO-68K

Stationery PRO-68K







TC

CARD PRO-68K

DATA PRO-68K

自由なレイアウト機能や入力するワープロ機能を装備し、カード型データベース

TO

CARD PRO-68K

BUSINESS PRO-68K

アートツール

NEW PrintShop PRO-68K

グラフィックライブラリ VOL.1

グラフィックライブラリ VOL.2

通信ツール

Communication PRO-68K ver2.0

Communication PRO-68Kのバージョンアップ版です。MNPモデムへの対応で、ハードフロー制御(CTS RTS)をサポートしています。



開発ツール

SX-WINDOW ver1.0
■CZ-258SS



マルチタスク環境、リアルタイム環境を生かし、使いやすく構築的なOS環境を提供します。

Human68k ver2.0

システムパフォーマンスをさらに高める処理機能を付加したHuman68kの最新バージョンです。

THE ver2.0

サウンドツール

Musicstudio PRO-68K ver1.1

MUSIC PRO-68K [MIDI]

Sampling PRO-68K

SOUND PRO-68K

MUSIC PRO-68K



























SHARP

敢えてX68000の大容量メディア対応を実証する意味。



X68000 SUPER登場

このたび新たにラインアップされた"X68000 SUPER"も すでに発売されている"SUPER HD"と同様 SCSIインターフェイスを標準装備しています また その他のシリーズには オプションとしてSCSIボード（CZ-6BS1）がサポートされ 大容量外部記憶装置をはじめ 各種SCSI装置との接続が可能になったのは ご存じのとおりです

SCSI規格とは……

SCSIは1986年にANSI（米国規格協会）で規格化された仕様で Small Computer System Interface の略 パーソナルコンピュータ



の周辺機器接続のための世界共通の規格です 大容量外部記憶装置（大容量ハードディスク、CD-ROM、DATなど）に加え 登場が期待される高速スキャナ 次世代プリンタなどのSCSI装置を デイジーチェーン方式で最大7台まで接続可能 大容量データの高速転送、および単一のインターフェイスでの周辺機器の複数制御が特長です

X68000と大容量メディア

ホビーとクリエーションのコンピュータグラフィックス X68000の本領ともいうべきこの領域は 感性あふれるユーザーにとって最も魅力的である反面 表現の緻密さに比例して必要な外部記憶容量も増大します サンプリング MIDI レイトレーシング その潜在能力をフルに引き出すには 大容量メディアへの対応が必須です。たとえば 新発売の光磁気ディスク（CZ-6MO1）と光磁気ディスクカートリッジ（JY-701MPA）なら、ディスク1枚で65,536色画像として1,000枚強 15.6kHzの音声サンプリングデータで約 時間強の情報を記録することが可能になる 「画像データ」 「オペレーションファイル」さえも SCSIに対応 夢の大容量メディア時代 です

X68000の先見性

初代X68000は すでにハードディスクインターフェイスを内蔵していたこと 当時まだ一般的ではなかったハードディスクに対して先見の発想で臨んでいたわけです 今回のSCSI対応も同様 100MBを超える大容量メディアへのエントリングがスタンダードになる日もそう遠くはありません

大容量ハードディスクか 光磁気ディスクか？ それとも……

考えもしなかった新しいデバイスか。新製品X68000 SUPERのSCSIインターフェイスに何を接続するかは 賢明なユーザー諸兄にお任せするとして このマシンがまた新たな一歩を踏み出したことに異論はないはずです 蛇足ながらこのSUPERシリーズに関していわせてもらえれば その日から大容量エントリングをお望みの方にはSUPER HDを、未来に夢を託すユーザーはSUPERといったところでしょうか

■デイジーチェーン方式による接続例



SCSIボードCZ-6BS1▶
標準価格29,800円（税別）



カラー画像ファイル、サンプリングファイルへ。
X68000のクリエイティブユースに待望の大容量ファイル。
書き換え可能な光磁気ディスク、登場。

■光磁気ディスクユニット■
CZ-6MO1・標準価格450,000円（税別）
■光磁気ディスクカートリッジ■
JY-701MPA・標準価格30,000円（税別）



NEW

SX-WINDOW, SCSIインターフェイス標準装備

X68000
PERSONAL WORKSTATION
SUPER



SUPER

EXPERT II

PRO II

充実のディスプレイラインアップ
DISPLAY LINE UP


Xstor
SHARP X68000専用ハードディスク
Xstor40はシャープX68000専用に開発されたハードディスクです
HXD40(外付2台用モデル)
Humanシステム(Ver1)以上。
¥118,000
¥188,000
¥98,000
ITEM
株式会社アイテム


レイトレーシングソフト
C-TRACE+、新発売。
メタボール対応
メタボール…
有機的な質感を表現する事ができます。
他のプリミティブと論理演算ができます。
αチャンネル機能…
αチャンネルへの対応により、
高度な合成作業が可能となります。
スコープ機能…
一度生成した画像の部分修正が
可能となります。
アニメーションへの応用により、
作業の大幅な効率化が図れます。
スポット光源への対応…
照射範囲の設定が自由にできます。
さらに照射範囲の境界のぼかしも
可能です。
DOS-EXTENDERへの対応…
CPUがV30のユーザーの方も
使用できるようにノーマル版も
バンドリング。(98版のみ)
対応機種…
NEC PC9801シリーズ
EPSON PC286シリーズ
EPSON PC386シリーズ
シャープ X68000シリーズ
定価…
¥198,000
※ワイヤフレームで物体を
確認できるワイヤ・ビュー
搭載予定。
レイトレーシングを高速にしたい方へ。
C-TRACE TP Ver 3.0 ¥298,000
C-TRACE98 EXTENDER ¥128,000
C-TRACE98 Ver 3.0 ¥98,000
C-TRACE68 Ver 3.0 ¥98,000
C-TRACE TOWNS ¥68,000
C-TRACE NEWS Ver 3.0 ¥630,000
★C-TRACE98 TP Ver 3.0 ¥298,000
★C-TRACE68 TP Ver 3.0 ¥298,000
CasT

第2回サイクロンCG大会記念ビデオ発売!!

応募作品紹介 | 実際利用法 | テクニカル解説 | 商品情報 価格¥3,000(税別)



サイクロンExpressα[エクスプレスアルファ] 好評発売中!!

3D・CG レイトレーシング サイクロンがパソコンCGの世界を越えた

● TOWNS版—「Z's TRIPHONY DIGITAL CRAFT」よりのデータコンバータ 12月発売予定

サイクロンExpressα98 (NEC PC-9801VX,RX,RA)	165,000円より
サイクロンExpressα68 (SHARP X68000)	98,000円
サイクロンExpressα TOWNS (富士通 FM TOWNS)	98,000円

サイクロンExpressαはキャノンの高画質カラーステーション(ピクセル・ディオ)に対応!!



彩ネット(BBS)開局!!会員募集中

株式会社アンス・コンサルタンツ



新たな挑戦の幕開けが迫る!

12月21日発売



続 ダンジョン・マスター カオスの逆襲

●X68000対応 ¥9,800(税抜)



ゲーム性、グラフィックス、サウンド何もかもがX68000の限界を超えた!!

好評発売中!

本格的3D快感シューティングゲーム ニューラル・ギア

●X68000対応 ¥6,800(税抜)

発売 ビクター音楽産業株式会社


X68000
アーケードフィールド宣言!
ATOMIC
アトミック・ロボキッド
ROBO-KID
全方向スクロール
ロボットシューティングの極限!
アーケード版アトミックロボキッド、X68000に登場!!
'90年年末、ついにその全貌を現わす。
新発売!!
2XXX年、核戦争の難を逃れた少数の人々は、地下に隠れて放射能の嵐がやむのを待ち、地上に出た。しかし、人々のDNA(遺伝体)は、わずかに残った放射能によって破壊され、人類は自分達の子孫を残せなくなってしまう。トミタ博士は、わずかに残された人々をシェルターに冷凍保存しDNA正常化のプログラムを開発する。ところが、シェルターに向かうロボキッドが動き出す直前に博士は、死んだ。自分の目的も解らずに、目覚めたロボキッド……はたして、ロボキッドは人類を救う事ができるのか?
ロクインNo.23のP170~P173に「徹底解剖!!」が掲載されているので注目!!
MIDI対応
X68000
5"2HD
標準価格8,800円
ジェミニウィング
Gemini Wing
アトレナリン全開シューティング
ジェミニウィング
大ヒット驀進中!!
スクランブル
ガンシップ緊急発進!!
ROUND2
ツインギドル
ROUND4
絶海の怒り
バルファ
ROUND5
スカルフェルド
MIDI対応
X68000
5"2HD
MIDIインターフェイスボードCZ-6BM1
標準価格8,800円
copyright ©TECMO
information
「サコムタイムズ」が発行されます。
下記の住所に
〒130
システムサコム
SACOM
株式会社システムサコム
ソフトウェア


宇宙は、野望だけでは支配できない。
宇宙暦796年、銀河系はゴールデンバウム王朝が支配する銀河帝国と、その専制政治に反対する自由惑星同盟の両陣営が激しい戦闘を繰り返していた・・・。圧倒的支持を得た「銀河英雄伝説」を遥かに凌ぐスケールで、今新たな伝説が生まれようとしている。銀河英雄伝説IIだ。銀河帝国の若き天才ラインハルト、そしてヤン・ウェンリーの熱い戦いが、再び始まる。星系マップは従来の4倍。3Dグラフィックによる戦闘シーンなど、あらゆる面でパワーアップされている。田中芳樹原作の大人気スペースオペラ「銀河英雄伝説」。宇宙の歴史を変える戦いは、ここに始ろうとしている。
SPACE WAR SIMULATION
銀河英雄伝説II
銀河英雄伝説II X68000シリーズ 絶賛発売中！ ¥9,800
●5.2HD(4枚組) ●X68000専用グラフィック ●2重スクロール ●MIDI音源対応 ●増設RAM対応
●FM音源・ADPCM対応 ●HMS(HYPER MOUSE SYSTEM)搭載
BOTHTEC ボーステック・ソフトウェア
株式会社クエスト 〒158 東京都世田谷区用賀2-18-8 TEL.03-708-4711
スタッフ募集 ゲーム企画・プログラマー・営業担当
TEL.03-708-4712


ACTION GAME
DYNAMITE DUKE
ダイナマイト・デューク
奴の野望は、俺が打ち砕く.!!
好評発売中!!
X68000専用
5' 2HD 3枚組 ¥8 800(税別)
ヘルツ
STAFF募集!!
プログラマー、デザイナー、サウンドクリエイター


ARTDINK
栄冠は君に
高校野球全国大会
X68000
絶賛発売中
全国3,990校の頂点を極めるのは、君だ。
アートディンク



X68000
12月21日
新発売!!
幽霊屋敷に消えたゴーストハンターたち…
本格ホラーRPG ゴーストハンターシリーズ#1
ラプラスの魔
原作
安田 均
音楽
小坂 明子
標準価格
8,700円
ユーザーズテレホン
Humming Bird Soft
株式会社エム・エー・シー ハミングバードソフト



怪魔退散
neXt
UNDEADLINE
幻獣鬼
X68000
5"2HD・3枚組 標準価格 ¥7,800
ACT-neXt
好評発売中
Rune Worth
黒衣の貴公子
RPG
ルーンワース 黒衣の貴公子
X68000
PC-9801VM UVシリーズ
PC-8801SRシリーズ VA
各¥8,800
発売待機中
遥かなるオーガスタ
SLG・neXt
遥かなるオーガスタ
T&Eの最新情報がわかるテレフォンサービス
Phone 052-776-8500
Technology & Entertainment Software
T&E SOFT
株式会社 ティーアンドイーソフト


X68000シリーズに、新登場！
シュヴァルツシルト
12月7日発売。
ストーリィ性を持ったドラマティックなゲーム展開
究極のゲームシナリオ
Schwarzschild
●5"2HD・3枚組¥12,800(価格には消費税は含まれておりません)
KOGADO
TEL. 03-353-7724


全国通販
SHARP 認定 PPO SHOP
O.A.ランド
(TEL) 03-770-8855
■アフターサービス万全のサポート体制
■TEL FAXのお見積OK!!
▶12・15～1・14
SHARPのことなら なんでもおまかせ.!!
SHARP X68000シリーズセット
X68000 SUPER NEW
X68000 SUPER HD
X68000 EXPERT-II
X68000 EXPERT II-HD
X68000 PRO II
X68000 PRO II-HD
周辺機器コーナー
OAランド特選品.!!
新製品 周辺機器
■光磁気ディスクユニット
■CZ-6MO1
特価¥450,000
■SCSIボード
特価¥20,000
■X-BASIC to C
CHECKER PRO68K
特価¥9,800
OAランド推奨 周辺機器
■SX WINDOW
特価¥5,100
特価¥15,800
■CZ-245LS
特価¥34,000
■CZ-8NJ2
▶¥18,000
■SX-68M
通信販売のご案内
全国通販
■12月より年内無休!!
㈱オー・エー・ランド
(03)770-8855
FAX (03)770-7080
■表示価格は、税別表示です。詳しくは、お電話にて、お問い合せ下さい。掲載の価格は、11月下旬現在です。

ちゃんとパソコンしたい人の
TSUKUMO

もう、離れられなくなるね。



あなたに、会いたい。

12月15日(土) **OPEN** AM10:15

うわさのパソコンロフト「ツクモパソコン本店」オープン

日本で初めての総合カメラ専門店「ツクモAV カメラ館」はオープンセール中

PRESENT ツクモ全店で1万円以上お買い上げの方先着1万名様に観月静香チャンのフロッピーカレンダーをさしあげます

静香チャンに会おう! 12月24日(月)PM1:00～パソコン本店3Fのイベントフロアにて観月静香チャンのサイン会が開かれます。

ツクモパソコン本店 〒101 東京都千代田区外神田1-9-7 ☎03-253-5555

ツクモAV/カメラ館 〒101 東京都千代田区外神田1 ☎03-254-3868

A Happy New Open!

ツクモAV カメラ館とツクモパソコン本店はX68000の聖地!どうぞおこし下さい。

掲載商品2万円以上

ツクモの3つのNEW 1 新しい年! 2 新しい店! 3 新しい商品!

シャープX68000コーナーが更に充実!! ツクモパソコン本店にX68000コーナーが移転し、4倍の広さに拡大!! 今まで以上にソフトからハードに致るまで更に充実 是非お立ち寄り下さい。

X68000シリーズ



新製品 **X68000 SUPER** CZ-604C SCSIインターフェイス内蔵タイプ

PRO II CZ-663C / CZ-653C

EXPERT II CZ-663C / CZ-653C

SUPER HD CZ-623C

本店オープン記念特価にて提供中!! 是非、お尋ねください。

年末年始 各種プリンタ大奉仕!

一流メーカー品

★48ドット熱転写カラープリンタ Happy特価¥59,800

★ドットマトリクス漢字プリンタ(80桁) Happy特価¥39,800

★ドットマトリクス漢字プリンタ(136桁) Happy特価¥59,800

パワーアップアイテム! 更に安く強力に! ハードディスク

SASIハードディスク

アイテック IT X640(40MB) Happy特価¥54,800

アイテック IT X680(80MB) Happy特価¥99,800

SCSIハードディスク

アイテック IT X80S(80MB) Happy特価¥99,800

アイテック IT X130S(130MB) Happy特価¥125,800

ハードディスク

40MB SASIタイプ Happy特価¥59,800

光磁気ディスクユニット

ソニー NWP-521N(光磁気ディスクユニット)

シャープ CZ-6SS1(SCSIボード)

SCSIケーブル

光磁気ディスク サービス

Happy特価¥408,000

ミュージックツール

Aセット ■CM-32L ■SX-68M ■Musicstudio Mu-1 Happy特価¥88,000

Bセット ■CM-64 ■SX-68M ■Musicstudio Mu-1 Happy特価¥138,000

アートツール

[ハードウェア]

CZ-6VT1

[ソフトウェア]

CANVAS PRO-68K ツクモ特価販売中!!

Z's STAFF PRO-68K Ver2.0 Happy特価¥49,300

マジックパレット Happy特価¥14,260

クロノExpress Happy特価¥13,300

デジタルクラフト Happy特価¥32,800

情報ツール

電子手帳シリーズ

PA-8500

PA-8000

CE-300L

電子手帳対応ソフト

■CYBER NOTE PRO-68K

■Stationery PRO

ツクモグローバルカード

メモリーボード

1MB増設RAMボード Happy特価¥19,000

2MB増設RAMボード Happy特価¥37,000

4MB増設RAMボード Happy特価¥64,000

コミュニケーションツール

■Communication PRO-68K Ver2 Happy特価¥15,000

ビジネスツール

■Hyper WORD

■CARD PRO-68K

■Kamikaze Happy特価¥37,800

開発ツール

■C Compiler PRO-68K Ver2.1

■SX WINDOW

■XBAS to C CHECKER PRO-68K

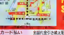
ポケットコンピューター

PC-E550

各種・各社新製品も大特価販売中! 詳しくはお問い合せ下さい。

ツクモ通販センター フリーダイヤル受注専用 **0120-377-999**

商品についてのお問い合せは各店販売又は☎03(3251)9911へ。



秋葉原各店 AM10:15～PM7:00

ツクモは「スーパーX PRO SHOP」です。

PRO STAFF ツクモ

便利で安心な通信販売 ツクモ通販センター☎03-251-9911

■ツクモAV/カメラ館

■ツクモニューセンター店

カード払い	全国代金引き換え配達	クレジット払い	現金書留払い	銀行振込払い	各種リース払い
	お申し込みは☎03-251-9911へお電話1本! 配達日の指定もできます	月々¥5,000以上の均等払いも	〒101 東京都千代田区神田 ツクモ通販センター		くわしくは係員にお問い合せ下さい。ケースに合わせてご相談にのらせて頂きます

超低金利クレジットOK!!回~84回払いまでOK!!　★頭金なし!★即日発送

またまた秋葉原でおなじみのP&Aがズバリ超特価セールでご奉仕!!

注目!!
平成3年2月末一括払い
手数料(金利)無料

全国通販

12/15~1/14

●お近くの方はお立寄り下さい。専門係員が説明いたします。
●本体単品で特価で受付します。詳しくは電話にてお問合せ下さい。
●ビジネスソフト定価の20%引きOK! TELください。

プリンター ■CZ-8PK ¥49,800

CYBER STICK 超特価!! ¥18,500

X68000シリーズ専用 MIDIインターフェースボード SX-68M(サコム) 特価¥16,480

X68000用メモリーボード(I/O DATA)(送料¥500)
1 PIO-6BE1-A ¥18,000
2 PIO-6BE2-2M ¥36,000
3 PIO-6BE4-4M ¥63,000

ジョイスティック 送料¥500
●X1PRO
●ASCII STICK

NEW X68000 SUPER/SUPER-HD/PROII/PROII-HD (送料・消費税込)





セットでお買い上げの方に
●ディスケット10枚
●ジョイカード2ケ
プレゼント中!!
○電話にて、ドンドンお問合せ下さい!!

SUPER
セットでお買い上げの方に●ディスケット10枚●ジョイカード2ケプレゼント中!!

Aセット CZ-604C-TN+CZ-606D-TN 定価¥427,800▶特価 価格はTEL下さい

12回	26,100	24回	14,300	36回	10,300	48回	8,100	60回	6,800

Bセット CZ-604C-TN+CZ-613D-TN 定価¥483,000▶特価 価格はTEL下さい

12回	31,800	24回	16,800	36回	11,700	48回	9,100	60回	7,700

SUPER-HD
セットでお買い上げの方に●ディスケット10枚●ジョイカード2ケプレゼント中!!

Aセット CZ-623C-TN+CZ-606D-TN 定価¥577,800▶特価 価格はTEL下さい

12回	37,900	24回	20,000	36回	13,900	48回	10,900	60回	9,100

Bセット CZ-623C-TN+CZ-613D-TN 定価¥633,000▶特価 価格はTEL下さい

12回	41,600	24回	22,000	36回	15,300	48回	11,900	60回	10,000

PROII
セットでお買い上げの方に●ディスケット10枚●ジョイカード2ケプレゼント中!!

Aセット CZ-653C+CZ-606D 価格はTEL下さい

12回	22,800	24回	11,800	36回	8,300	48回	6,800	60回	5,400

Bセット CZ-653C+CZ-605D 価格はTEL下さい

12回	12,800	24回	13,800	36回	9,000	48回	7,100	60回	5,900

Cセット CZ-653C+CZ-606D 価格はTEL下さい
Dセット CZ-653C+CZ-613D 価格はTEL下さい
Eセット CZ-653C+CU-21HD 価格はTEL下さい

PROII-HD
セットでお買い上げの方に●ディスケット10枚●ジョイカード2ケプレゼント中!!

Aセット CZ-663C+CZ-605D 価格はTEL下さい

12回	30,800	24回	16,300	36回	11,300	48回	8,800	60回	7,400

Bセット CZ-663C+CZ-606D 価格はTEL下さい

12回	32,500	24回	17,100	36回	11,800	48回	9,300	60回	7,800

Cセット CZ-663C+CZ-606D 価格はTEL下さい
Dセット CZ-663C+CZ-613D 価格はTEL下さい
Eセット CZ-663C+CU-21HD 価格はTEL下さい

X68000シリーズ ~P&Aスペシャルセット=限定読上販売!!

台数限定　送料・消費税込み　※セットでお買い上げの方に、●ディスケット10枚●ジョイカード2個プレゼント中!!

EXPERTII
Aセット P&A厳選セット
■CZ-603C
■CZ-606D
▶P&A超特価 ¥304,000

Bセット ■CZ-603C+CZ-604D ▶特価¥309,000
Cセット ■CZ-603C+CZ-606D ▶特価¥322,000
Dセット ■CZ-603C+CZ-613D ▶特価¥342,000
Eセット ■CZ-603C+CU-21HD ▶特価¥347,000

EXPERT-HD
Aセット P&A厳選セット
■CZ-612C(ブラック)
■CZ-606D(ブラック)
▶P&A超特価 ¥335,000

Bセット ■CZ-612C+CZ-604D
Cセット ■CZ-612C+CZ-606D
Dセット ■CZ-612C+CZ-613D
Eセット ■CZ-612C+CU-21HD

■モデム 限定
PV-A24VM5(アイワ)
●MNPクラス5
特価¥29,900

■ALL in Note
フリートップ パーソナルコンピュータ
AX-286N-H2
P&A超特価 価格はTEL下さい

■本広告の掲載の製品の価格については、消費税は含まれておりません。●営業時間=平日AM10 00~PM7 00 日祭AM10 00~PM6 00

X68000用ソフトコーナー(送料1ヶ~5ヶまで¥500)

X68000用ハードディスク(送料¥1,000)

アイテム		
●HXD-040(40MB/23ms)	定価¥118,000▶	特価¥88,000
●HXD-042(増設用)	定価¥128,000▶	特価¥95,000
アイテック		
●ITX-640(40MB/28ms)	定価¥156,000▶	特価¥83,000
●ITX-680(80MB/20ms)	定価¥156,000▶	特価¥97,000

プリンター(ケーブル・用紙付)限定5台 新品(送料¥1,000)

■CZ-8PC5(カラー漢字24ドット熱転写プリンター) 定価¥65,800 特価¥39,800
■CZ-8PK9(24ピン漢字プリンター136桁) 定価¥152,000 特価¥49,800
●CZ-8PC4 P&A特選!!(カラー漢字48ドット熱転写プリンター) 定価¥99,800 特価¥57,000

周辺機器コーナー(送料¥500)

モデムコーナー (送料¥1,000)

P&A特選パソコンラック (送料無料)

A 3段 875(H)×580(D)×610(W) ¥9,000
B 4段 1320(H)×600(D)×630(W) ¥11,500
C 5段 1280(H)×600(D)×620(W) ¥15,000

中古パソコン(セットはモニター付)送料¥2,000

中古パソコンはP&Aにおまかせ!!

便利な超低金利クレジットをご利用下さい
■月々¥1,000円からOK!! ■ボーナス払いOK(夏冬10回までOK)
■支払い回数 1回~84回 ■お支払いは3ヶ月先からでもOK!!

アフターサービス万全

通信販売お申し込みのご案内

■現金特別価格でクレジットが利用できます。残金のみに金利がかかります。



P&A 株式会社ピー・アンド・エー
〒124 東京都葛飾区新小岩2丁目1番地19号
☎03-651-0148(代) FAX 03-651-0141
●マイコン ●ビデオ ●ビデオテープ

■現金書留及び銀行振込でお申し込みの方は上記商品の料金に3%加算の上でお申し込み下さい。詳しくは、お電話でお問い合せ下さい。

P&A超低金利クレジットをご利用ください!!

超特価でクレジットが組める!!

謹賀新年PRO-68K

あけましておめでとうございます。では，いきなり付録ディスクのお披露目といきましょう。
これが今回のディスクに詰め込まれたプログラムの数々です。詳しい解説は本文をどうぞ。









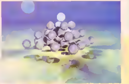
今回のディスクの目玉となるVS2.Xです ビジュアルシェル上でテキストやグラフィックの表示を可能にするものです





……4枚までのグラフを扱えます 簡易グ……クエディタ機能つき。

柔らかな影を持つレイトレーシングツールKASHIによる画像の作成画面 光源からの……









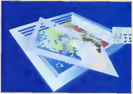
コン……CARD2.FNC これはカードキャラクタの原画 256色データにも対応しました 花札だろうが モンメカだろうが なんでもこなせます



カードゲーム……NEKE 有名なトランプ1人遊びです ……では……されるゲームです







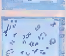




Z's STAFF Kid68の機能を！ 91年8月号で扱ったフラクタル処理，……のようなフレア処理，3次元マ……などの例。







ディスクに含まれるSX-WINDOW用アクセサリ。SX-WINDOWでお馴染みの目玉からライフゲーム，パス名取り込みツールなど。

MOの時代がやってきた

光磁気ディスクシステムCZ-6MO1

シャープ純正CZナンバーの光磁気ディスクドライブが発売され、X68000の環境が大きく鍛える価格ではないとはいえ、ここはひとつ皆さんのふところと神田を逆撫でしつつその概略を紹介しよう。

Tan Akihiko 丹 明彦

僕は現在、編集部マシン室のX68000 SUPER HDをほとんど私物化している。その僕は今、嬉しくてしかたがないのだ。SUPER HDの80Mバイトに飽き足らず今度は光磁気ディスクに自分の環境を作り上げようとしているのだから。そってきる日がとうとうきたのだから。

ディスク容量不足に嘆くユーザー待望の光磁気ディスクシステムがついに発売された。しかも45万円という、かなり思い切った価格設定である[2]。これでX68000の環境がまた一歩広がることになる。現在、光磁気ディスクは編集部で稼働している。今月号の付録ディスクの原版も光磁気ディスクに格納してある。

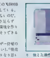
■製品概要

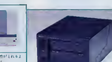
すでにご存じのとおり、光磁気ディスクは、パーソナルコンピュータなどの大容量外部記憶装置のひとつである。

原理的にはハードディスクよりもむしろCD-ROMに近いのだが、使う上では、大容量のハードディスクという感覚である。なぜなら、光磁気ディスクは、読み出しだけでなく書き込みもできるからである。光磁気ディスクは、この点でCD-ROMよりも大きくポイントを稼いだ。ただ、これでCD-ROMが劣っているというのは筋違いである。CD-ROMと光磁気ディスクは用途が違うというふうにとらえるのが正しい。

光磁気ディスクユニットCZ-6MO1はSCSIインタフェイスを介してX68000と接続するようになっている。1990年に出たX68000SUPER HDはSCSIインタフェイスを標準装備している。これは、今回の光磁気ディスクもそうだが、これからの外部記憶の標準はSCSIになるだろうと見越してのことであろう。もちろん、SUPER HD以外のユーザーでも光磁気ディスクを使用することはできる。オプションのSCSIボードを利用すればよいのだ。ただし、拡張スロットがひとつつぶれる。

次に、記録フォーマットであるが、ISO規格に準拠したものとなっている。これによ





り、他との互換性がとれるようになっている。

光磁気ディスクのメディアは取り外し可能である。ディスクはガラス製で、さすがにCD-ROMのように丸裸とはいかず、3.5インチディスクを思わせるプラスチック製のカートリッジの中に収まっている。カートリッジは、音楽用CDのケースくらいの大きさである（光磁気ディスクは5.25インチ）。カートリッジ表面をしげしげと眺める。A面やB面などとある。そう、光磁気ディスクは両面使えるのだ。よく見るとライトプロテクトタブもついている。ライトプロテクトのかかるハードディスクはそれほどないことを考えると、これは実にありがたいことである。

そのままでは残念ながらディスクの顔を拝むことはできない。ディスクはカートリッジの中に隠れているのだ。だから些細なことでは汚れないようになっている。しかし、ここではちょっと失礼して、カートリッジのシャッター（扉のことね）を開けてみる。ちょっとCDに似た、しかしずっと色鮮やかな虹色を見ることができる。ディスクがガラスでできているにしてはカートリッジは意外に軽い。しかし、気軽に構えるのはその値段を知るまでのこと。カートリッジは1枚30,000円。これを聞けば、今まで軽かったカートリッジが10倍くらいの重さに感じられることだろう[1]。

手のひらで弄ぶのもいい加減に飽きてきたのでカートリッジを光磁気ディスクユニットに入れる。すると、ビデオデッキのように、オートフロントローディングをしてユニット内に収まる。少し（数秒）待つと使えるようになる。取り出すときはイジェクトボタンを押す。若干の時間をおいて（処理の後始末をしているのだろう）、やはりビデオデッキのようにカートリッジが吐き出されてくる。

最大のウリであるディスクの記憶容量であるが、両面で594Mバイト。一度に使えるのは片面だけ[2]なので、実質300Mバイト弱だが、それでも十分すぎる容量といえるだろう。書き換え回数100万回以上（つまり、めったなことでは限界に達しない）。これで1枚30,000円なら十分に安いといえるだろう。

光磁気ディスクのメディアも別売りである。何枚でも買い足せるのだから当然である。光磁気ディスクは、フロッピーディスクの持つ便利さとハードディスクの持つ便利さの両方を合わせ持った、魅力的な記憶装置であるといえよう。

ここで、光磁気ディスクシステムを稼動するのに最低限必要なものを並べておく。定価は税別である。

- 光磁気ディスクユニット
  CZ-6MO1　450,000円
  光磁気ディスクカートリッジ
  JY-701MPA　30,000円
- SCSIボード（SUPER HDには不要）
  CZ-6BS1　29,800円

当然、OSもSCSI対応にしなくてはならないが、これはSCSIボードにソフトウェアが同梱となっているので、特別に揃える必要はない。

■使ってみる

一般に光磁気ディスクをHuman68kで扱うためには、SCSIデバイスドライバであるSCSIDRV.SYSを組み込んでおく（CONFIG.SYSに書いておく）必要がある。これを忘れると、せっかくつないだドライブが認識されないという間抜けな事態に陥るので注意。

初期設定は通常のハードディスクと同じ。まず初期化および領域確保をして300Mバイト近い領域にいくつかのパーティションを切る。1面まるごと1ドライブというのも、豪快でいいかもしれない。

使ってみるといっても、より大容量になったハードディスクという以外の感想はない。ただ、この大容量というのがとてつもなくありがたいのだ。画像ファイルをたくさん転がしておくのは、たとえハードディスクといえども犯罪行為に近いものがある。その点光磁気ディスクなら、巨大な倉庫として使うこともできる。PICファイルを収集して1カ所に置いておくのもよし、サンプリング音をまとめてライブラリにしてしまうもよし。ハードディスクでもかなりつらいTeXのファイルをしまい込んだところで、痛くもかゆくもない。しかも、これらがいつでも利用したいときに素早く利用できる。まったくこたえられない。

速度の点はやや心配されていたが、使っている上では特に遅いと感じない。十分にハードディスクの代わりができることと思う。具体的な数字を挙げておくならば、

- 平均データ転送速度（読み出し時）
  635Kバイト/sec
- 平均シークタイム
  105ms

ということである。単純な数値比較はできないが、ハードディスクと比べてもそれほど遜色ないはずだ。実際、いろいろなプログラムで試してみると、確かに多少余計に時間はかかるが、その差はほんのわずかなものである。

動作音は、ハードディスクに比べるとやや気になる。それでも、ソニーの光磁気ディスクに比べると相当静かだというのは、ソニーの光磁気ディスクドライブを実際に使っている祝一平氏の弁。もちろん、昔のハードディスクに比べるとはるかに静かである。

SX-WINDOWは光磁気ディスクにしっかり対応していて、ドライブトレイには「MOドライブ」の文字が出てくる。MOとはMagneto Optical diskの略で、要するに光磁気ディスクのことを指す。逆にビジュアルシェル（バージョン2）は光磁気ディスクだけでなくSCSIデバイス全般を認識しない[4]。

◆

画像が大量に置けるのは嬉しいことである。CGをやっていると、なにかと画像用のファイルがたくさんできる。出来上がった画像はPICで圧縮してしまえばまだいい。しかし、マッピングに使うような大きくてなおかつ圧縮のできない（圧縮するわけにはいかない）画像ファイルはそうそう長いこと転がしておくわけにもいかない。100Mバイト単位で気兼ねせずに使えるというのはまさに夢のような環境。通信で絵や音楽のデータをいっぱい抱え込んでいる人にもおいしい。ほしいデータをフロッピーをとっかえひっかえして探す悪夢から解放されるのだから。

ただ、画像をもっと大量に扱っていると、光磁気ディスクといえどもさすがにあふれることもあり得る。とくにアニメーションをするようになったらそうなる可能性は大きい。そのときは新しくメディアを買うのだ。1作品1枚なんというのも辞さない。フロッピー600枚に匹敵する容量のディスクが、フロッピー300枚ぶんの値段で手に入ると思えば安いもの。しかもパフォーマンスはフロッピーをはるかに上回る。

ちょいと初期投資は大きいが、後々までおいしく使える光磁気ディスク。シャープ純正だからX68000にも安心して使える。今の環境でハードディスクが狭いとお感じの方は、買い替えを真剣に検討すべきなのかもしれない。

SCSIボードCZ-6BS1について

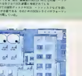


光磁気ディスクと同時に、それをSUPER HD以外のX68000と接続するためのSCSIインタフェイスボードもシャープから発売された。

SCSIとはSmall Computer System Interfaceの略で、パーソナルコンピュータと周辺装置との間でデータをやりとりするための標準規格である。その周辺装置は、主に大容量ハードディスクや今回紹介した光磁気ディスクといった外部記憶装置である。

X68000のSCSIインタフェイスは、SCSI規格の外部記憶装置を7台までデイジーチェーン接続することができる。現在、編集部のSUPER HDにも内蔵ハードディスクと光磁気ディスクの合計2台のSCSI装置が接続されている。

SUPER HD以外のX68000で光磁気ディスクやSCSIハードディスクなどを使いたい場合は、このCZ-6BS1が必要である。そのためのSCSIドライバやフォーマットなどのソフトウェアは付属している。



1）シャープは光磁気ディスクに関してはかなり早い時期に手をつけていたほうで、この製品も実は同社初のものではない。ただしシャープが以前に作った光磁気ディスクシステムは実に60万円もするしろものであったが……。

2）45万円という価格は、半年ほど前にソニーが達成（？）している。しかしいずれにしても、価格設定としてはかなりきついほうで、普及ラインには乗らないという声も聞く。

3）ディスクを入れ替えたり裏返したりすれば、ドライブの設定状態（つまり、AとBドライブがハードディスクでCが光磁気ディスク、というような）がまったく変わってしまうので、X68000本体をリセットして、システムを立ち上げ直さなくてはならない。頭がフロッピーディスクに似ているし、イジェクトもできるのだから、これはなにか不便なような気もする。しかし、光磁気ディスクの場合、メディアの入れ替えはハードディスクを載せ替えるのとおなじくらいおおごとなのだから、リセットする、とは必要である。第一、光磁気ディスクはフロッピーと違い、パーティションを切って複数のドライブに分けて使うようになっている（ハードディスクと同じ）。入れ替えたディスクのパーティションが前と異なれば、アクセスに混乱をきたすのは目に見えている。

しかし、欲をいえば、レーザーディスクのように両面連続使用をしたり、両方の面のドライブを同時にアクセスできたりするともっといいのかも。

4）というわけで、X68000SUPER HDにはビジュアルシェルは付属していないが、今月号の付録ディスクにはVS2.Xとそれを日本語化するVS2.Xが収録されている（もちろんSCSIにも対応している）のでSUPERのユーザーの方も探していただきたい。

DōGA・CGアニメーション講座

今月はアニメーション3作品と、DōGA開発のウィンドウシステム「KD-Window」の登場です。

まずは、モデラー高津の今月のアップデータです。これは、もしやあの"ウルトラホーク1号"では？ なんでもちゃんと3機に分かれるそうなんです。いやあ、凝ってますねぇ。

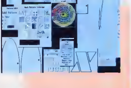
お次は本職の画家の方が作ったという「SWORD」です。なんとディスク5枚組だったというから、すごい大作ですね。これは技術的にも素晴らしいので、ぜひ参考にしてください。しかし、この画家さん、本職のほうは大丈夫なんでしょうか？

最後にDōGAのスタッフがしゃかりきになって開発しているウインドウシステム「KD-Window」の開発途中デモをお届けします。アプリケーションその他はモデラー高津のLOGINのほうで説明されていますので、そちらもご覧ください。





[illegible]ま開発中のDōGAオリジナルウィンドウシステム。CGAシステムVer.3は[illegible]で組む予定との[illegible]と





プリンス・オブ・ペルシャ
アメリカで大人気のアクションゲーム。ワナでいっぱいの迷宮を冒険し、ねくねの少年を操作し、1時間以内に姫を救い出せ！



SOFTWARE INFORMATION

あけましておめでとうございます。こうして1年を振り返ってみると、いやー、ゲームって増えましたねぇ。てなわけで、このページも今月号から内容が少し変わります。ぜひ感想を聞かせてね。

話題のソフトウェア

今月[illegible]ブローダーバンドの**プリンス・オブ・ペルシャ**[illegible]。このゲームはアメリカ製のアクションアドベンチャーで、本当に誰もがな動きと簡単操作、独特のグラフィックがウリ。X68000版ならグラフィックは[illegible]。AMIGA版で出力されているようなリアルな効果音も、ぜひ再現してもらいたいところです。発売日は未定だけど楽しみに待っていようね。

移植ものといえば、こちらもそう。ウェーブトレインの**スペース・ローグ**。これはAPPLE IIで人気だった同名ソフトの移植もの。[illegible]を舞台にしたシナリオと[illegible]ーンなんだけど、[illegible]の[illegible]画面がカラーと変わってRE[illegible]で、[illegible]度おいしいってカンじですね(ちょっと例が古かったかね)。これはすでに発売中。

さて、次はヘルツの**ダイナマイト・デューク**といきましょう。アーケードゲームの移植とあって、以前とは勝手が違うので発売がやや遅れていましたが、ようやくお目見えです。アクションとシューティングの要素を持つこのゲーム、[illegible]見の価値はありそうですね。こちらも発売中です。

さあ、X1turboユーザーの皆さん、お待たせしました。やっとあの**プール・オブ・レイディアンス**が発売されました。AD&D®の"フォーゴトン・レルム"を舞台にした本格ファンタジーRPG。作成からキャラクターまで凝りに凝ったんだけど発売は遅れちゃったけど、[illegible]〜るぞ、わあ！



シューティングゲーム、爆進!!

1	フグーン	
2	ナイアス	初
3	グロディウスII	1
4	ダンジョンマスター	6
5	ソムンティー	2
6	ノーサリアン	4
7	超戦王ルーエアー コンバット〜	7
8	ボビュラス	5
9	サイバリオン	3
10	イメージファイト	初

2カ月連続で1位はフグーン。しかし今月の話題はそんなことではない。なんとナイアスが初登場2位!! あのシェノサイトと互格のスタートをきったというから驚き。さっそく、ガキの声を聞いてみましょう。

「3面のマスタースクロールは凄い」「シューティングの進期になる」「ゲームバランスがとれている」「マスターが凄い」「グラフィックがいいわり」「ラスタースクロールがイカす」など、なんかマスタースクロールの話ばっかり(笑えてみたら約3割)。ま、新規参入なのに技術力がしっかりしてるというコトの現れなんでしょうけど。このいガキの勢いからすると、来月はひょっとしてひょっとするかもしれないぞ。さあ大変だズームファン。

ナイアスだけでなく、グロディウスIIイメージファイトなどほかのシューティングも続々ランクアップ。初登場だけで3位のグロディウスIIは、X68000とグラディウスシリーズとの関連もあって、デキ次第ではアフターバーナー並みのヒットもありえそう。イメージファイトはゲーセンで難しさにあきらめた人が再びチャレンジしているケースが多いようです。

アクション以外ではダンジョンマスターが唯一ランクアップ。着実な評価のわりに健闘しています。なお『カオスの逆襲』が検討中なので、ガキにはアプライトしまで書いてください。よろしく。

[illegible]のすぐ下で[illegible]ウイング、[illegible]などが踏ん張っています。この強力なメンバーによるバトルは、来月に向けてさらに激しさを増す…。[illegible]年始のランキングに乞うご期待！ (雪)

パロディウスだ!

早くもファンから熱い期待が寄せられている「パロディウスだ!」。今回も画面写真しか間にあわなかったので、これで我慢してちょうだい。

しかし、画面を見るかぎりホントに完璧なデキ。[illegible]ろもち細部がつまってるかなという気もするけど、でも本当にX68000の画面なのかどうか疑っちゃうくらいだ。カ[illegible]のおネエさんもモアイ戦艦もそのまんま。ファンタークグラフィックも再現されていて、このままスイ戦いちゃったらX68000の画面をまだまだ塗り替えるとんでもない作品になっちゃうかもしれない。

とにかく、大いにきょう体を揺らしたゲームセンターで[illegible]たソフトの[illegible]のデキは[illegible]なのであります。(ま)

X68000用　5"2HD[illegible]　価格未定
コナミ　☎03(262)9[illegible]

●画面は開発中のものです。



エメラルドドラゴン

みんなが待っていたエメラルドドラゴン。あれほど待ったユーザーもとうとうその時がやってきたのだよ。このエメラルドドラゴンというゲーム、プレイヤーは主人[illegible]ルーン[illegible]なって4人の仲間たちと聖ボーパーンをさまよい戦っていくというタイプのRPGなのだ。

とにかくこのエメラルドドラゴン[illegible]広い。誰も飽きさせない展開で[illegible]も、アニメ絵のタムリンもかあいい[illegible]本当にかあいいなど[illegible]。[illegible]88ユーザーの間では[illegible]のエメドラノンピまで出てしまったくらい[illegible]とんでもなく評判のよかった[illegible]。X68[illegible]版はそれよりさらにさらにグラフィックは[illegible]なり、音楽もX68000の機能に合わせて[illegible]パワーアップ[illegible]など(で)

X68000用　5"2HD[illegible]6枚組　9,800円(税別)
グローディア　☎03(220)5225



マーブルマッドネス


●画面は開発中のものです

ホームデータの手によって移植が進んでいる「マーブルマッドネス」の最新写真が届いたので紹介しよう。基本部分はもう完成しているようで、これから敵キャラやいろんな仕掛けを組み込むようだ。

マーブルマッドネスの"ノリ"の仕掛け。基本は、玉を転がして時間内にゴールまで運んでやるというものなんだけど、ハネ橋や波打つ床、突然現れて玉を飲み込もうとする掃除機まであの手この手で行く手を阻んでくるのだ。

このゲーム、[illegible]てはAMIGA版ですでに遊ばれ尽していて、わざわざトラックボールまで買ってくるほどの人気。画面を見るかぎり、移植はなかなか順調のようだから、X68000ユーザーが手にする日もそう遠くない。完成版が届き次第[illegible]紹介するのでお楽しみに。(湘)

X68000用　5"2HD[illegible]　価格未定
ホームデータ　☎078(261)2790

ザ・スーパーラスベガスII

[illegible]の旅行にはアメリカに行って、ついでにラスベガスで大金持ちになっちゃおうかなどと言っている部長もおそらく[illegible]人(推定)はいるかと思いますが、世の中そんなに甘くない。中は何人も[illegible]と対戦[illegible]まず、このスーパーラスベガス2で鍛えましょう。

ポーカー、セブンブリッジなどのカードゲームはもちろん、バックギャモンやルーレットなぜかチンチロリンまであります。ひとつのゲームで勝った分はほかのゲームに持っていくことができるので、ちょっと負けてもあきらめないでほかのゲームに挑戦してみるのもいいんではないでしょうか。グラフィックがなかなかアメリカンしていていいですぞ。[illegible]ラスベガスの[illegible]にはスロットマシンがないけど。

さあ、これさえあればあなたも明日から大金持ち気分。[illegible]のかわりにきつねそばが食えますぞ。(湘)

X68000用　5"2HD[illegible]7枚組　12,800円(税別)
日本デクスタ　☎03(833)47[illegible]



アステロイド・クイーン/JANJON デルタアーム/ニニンバトル

タケルを通して発売されているログイン誌の「ソフトウェアグランプリ」入賞作から、最近発売されたものをいくつかご紹介しよう。

まずグランプリ受賞作「アステロイド・クイーン」。マウスで自機を操作して、敵弾を交わし[illegible]UFOの母艦を破壊するというゲーム。マウスによる操作感覚がウリだとか。

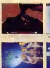
JANJONはアクションパズルゲーム。じゃんけんがモチーフで、敵も自分も手の形をしているのだ。自分の形を変えてしまうアイテムや、ある特定の形でないと通れないドアもあったりして、面をクリアするのに頭を悩みそう。

デルタアームは横スクロール型シューティング。しかし、自分でパワーアップアイテムを出せたり、やられ続けていると難易度がさがったりとなかなか斬新な試みが投入されている。パターン化が叫ばれるシューティングに一石を投じる作品となるか?

最後のニニンバトルはジョイスティック[illegible]をつなぎ、2人で対戦するアクションゲーム。[illegible]や能力の違うロボットのなかから好き[illegible]のを選んで戦うことができる。

いずれもアマチュアの作品だが、完成度はかなり[illegible]。2,800円という価格もうれしい。

アステロイド・クイーン　X68000用　5"2HD[illegible]
JANJON　X68000用　5"2HD[illegible]
デルタアーム　X68000用　5"2HD[illegible]3枚[illegible]
ニニンバトル　X68000用　5"2HD[illegible]
以上すべて2,800円(税込)
ブラザー工業(タケル)　☎052(824)2493



生中継68

コナミの生中継68の写真が大量に届きました。このゲーム、とにかく細部へのこだわりがすごい。バックネット、一塁三塁スタンドの構図は当たり前、ライト、センター、レフトスタンドの構図もあるし、風船飛ばしやプロ野球ニュースとそのオープニング( )まであるんです。これだけ大量の写真を送りたくなる気持ちもわかろうというもの。キャラクターはサイズも動きのいずれもいいみたいだし、2人プレイ時に専用の画面が用意されている(とおばしき)写真もあって、熱意のほどがうかがえます。

X68000用野球ゲームも再び元気を取り戻しそうで嬉しい限り。[illegible]の生中継68が、世界にコナミとX68000の力をガンガン見せつける一作になることを期待したいですね。(湘)

X68000用　5"2HD[illegible]　価格未定
コナミ　☎03(262)9[illegible]

●画面は開発中のものです



ラプラスの魔

X1やPC-88[illegible]などで先に出ていたので名前くらいは知ってるって人も多いラプラスの魔。[illegible]あたってちゃあーんとX68000のパワーに対応してオペレーション、ビジュアル、ミュージックすべてがリニューアルされているのだ。

このラプラスの魔はちょっと変わったタイプのRPG。ファンタジータイプのゲームとはちょっと違って、[illegible]への攻撃に[illegible]ではなく[illegible]が使えたりとか、[illegible]のにモンスターの写真を撮って[illegible]で売ったりと、なかなかにユニークなゲームシステムなのだ。そうそう、クトゥルー神話との関わりも深いという[illegible]だから[illegible]オタッキーには[illegible]たえられないかもしれない!?

現在のところ制作も順調。来月は完成版でのレビューができると言うのでお楽しみに。[illegible]下もロータス[illegible]とはいったんだ、ロータス[illegible]は!?)

X68000用　5"2HD[illegible]3枚組　価格未定
エム・エー・[illegible]ングパートソフト　☎06(3[illegible])8255



中華大仙



アーケードで出ていたTAITOの「中華大仙」がX68000で発売されることになりました。移植を担当したのはおなじみSPSということで出来のほうはますます[illegible]。このゲームはその名から連想されるとおり中国が舞台。中国というと西遊記、西遊記というと孫悟空、というわけでこの「中華大仙」は孫悟空が主人公の横スクロールシューティングゲーム。出てくるのは、ぎょうざ、シュウマイなどいかにも中国的な敵キャラ。背景は水墨画風。それに加えて中国風の音楽が聴けるとなると、これはもう中国人になりきってバンバンと撃ちまくるしかない。また、ボスキャラを倒しステージクリアすると「大きな仙人」がほめてくれる。うーん、とっても「中華大仙」。(丸太)

X68000用　5"2HD[illegible]　価格未定
シャープ　☎03(260)1[illegible]

栄冠は君に

高校野球を舞台に監督として手腕をふるうシミュレーションゲーム「栄冠は君に」が登場。

全国の高校を相手し勝ちぬかせてあるというヘビーそうなゲームだが、マウス操作のおかげでその辺の重さは感じさせずに自然にプレイできる。

でも、なにはともあれまず[illegible]。[illegible]の中から[illegible]選び、全国大会を目指して練習に練習を重ねていくのだ。1人ひとりに特別メニューを組んだりもできるし、[illegible]をつけさせたりもできる。とにかくここでは地道な努力(なにしろ[illegible]期間が40年)が必要。しかし、ここでちゃんと頑張った人には、必ず勝利の女神が微笑むはず。

さあ、甲子園目指してガンバ[illegible]。(湘)

X68000用　5"2HD[illegible]3枚組　9,800円(税別)
アートディンク　☎0474(77)754[illegible]



あのゲームミュージックフェスティバルがCD/ビデオになったって!? そりゃ買わなきゃすぐ買わなきゃ、はへはへ。

※ZUNTATAはTAITO、S.S.T.はSEGAのゲームミュージックスタッフの名称です。1990年8月25日、この2大ゲームメーカーによるライブが行われたのでした。

■ZUNTATA LIVE

ゲームミュージックのようなインスト(歌の入らない楽曲)のライブ音盤は大変だ。ボーカルのある曲なら視聴者は歌を追いかけて聴くため演奏者のミスは耳につかないものだが、インストの場合は演奏と演奏の音色のからみの心地よさを重視し音質のため、ひとつのトチリが大失敗につながることさえある。ZUNTATAはライブ演奏は初めてとの"と[illegible]。確かに演奏は緊張を隠せてなかったけど演奏はさすが、うまいうまい。[illegible]もいいし、アレンジもいい。「DADDY」のキーボードソロご[illegible]でした。こうして聴いてみるとTAITOのゲームミュージックって作りが実にライブ向きですねぇ。

■S.S.T. BAND LIVE

S.S.T.はライブ演奏を[illegible]すでに[illegible]しており、演奏や[illegible]が感じられる、バランスのとれた音の「はめかた」は、もはやプロそのもの。以前よりギターが前面に押し出され、とてもヘビーで迫力のある音になった。演奏曲目のほうはベストヒットの「AFTER BURNER」や新作「G-LOC」、またテーブル筐体の「ゴールデンアックス」、メガドライブの「ファンタシースターIII」とバラエティに富み、[illegible]アレンジで聞かせてくれる。

――ZUNTATAは[illegible]、S.S.T.は[illegible]アレンジ重視という[illegible]内容的には引き分けかな。次回[illegible]は私も行きたい。(参)



○サイトロンビデオコンサート

1月24日、[illegible]にて、先日行われたS.S.T.とZUNTATAジョイントライブのビデオコンサートとファンの集いが行われました。当日は多くのファンがつめかけ、上映が始まるやいなやみんな手拍子。「AFTER BURNER」ではライブそのままに立ち上がって[illegible]まで現れるテンションの高さ。終了後[illegible]S.S.T.とZUNTATAの[illegible]が現れてインタビュー。今後はもっとライブにも力を入れたいとのこと。ゲームミュージックも熱心なファンが増え、今後イベント活動も盛んになるはずです。ただ、コメントの最中にまで叫んでくる女性ファンがいて、メンバーも苦笑い。有名になるって[illegible]だなあ。(湘)



●ソル・フィース

多関節キャラの動きに注目だあ

ウルフの新作はサイドビュータイプのシューティングゲーム。オープニングはアニメーション、ゲームは多関節キャラの多彩な攻撃と、目にもうれしい作品だ。MIDIにも対応しているぞ。

Yamada Junji
山田 純二

最近 やけにシューティングゲームが多いなあ。先月号でも移植もの、オリジナル合わせて3本のゲームが紹介されていたし、さらには、「パロディウスだ！」までもが登場。なーんでまたこんなに集中しちゃってるんだろ。

で、ウルフ・チームからも完全オリジナル、バリバリのシューティングゲームが登場ってワケだ。これだけ数が揃っている（しかも一時期に集中している）ジャンルに挑んでいるのだから、この「ソル・フィース」に対するウルフ・チームの自信のほどがうかがえる。

さて、と。それでは、その自信がどこからくるのか、そして基本ともいえるサイドビュータイプのシューティングゲームを、ウルフがどうやってデザインしたか、じっくり見ていくことにしよう。

フルアニメーションだよん ◆◆◆◆◆◆

まずは、オープニングにおけるアニメーション。これは、全部で60秒くらいの全画面描き替えによるアニメーションだ。ふむ、X68000で見るアニメはなかなかの見ごたえだ。256×256モードの粗い画面ではあるが、そのぶん動きで見せてくれるので気持ちがいい。ラストの画面で思わずサ○ラ○ズのナョップを探してしまったほどだ。ゲームの本編とはまったく関係ない部分で



X68000用 5"2HD版3枚組 8,800円［税別］
ウルフ・チーム ☎03(5273)4781

こんなことをやっても不毛だ、と思う人もいるだろうが、個人的にはここまでこだわったのには好感が持てる。と、ここまでほめていながらいうのもなんだが、グラフィックはもう少し頑張ってほしかったな。

ちなみに、宣伝文句のフルアニメーションというのは、ディズニー映画などにおける1秒24コマの細かい動きという意味ではなく、オープニング全体を通してアニメーションしているといった意味のようである。

ここに注目！ ◆◆◆◆◆◆◆◆◆◆◆◆◆

いきなり、脱線してオープニングをほめてしまったが、もちろんゲーム自体も頑張っている。ゲーム構成は、さっきもいったようにサイドビュータイプのシューティングで、オプションによるパワーアップ、各面の最後にいるボスキャラを倒すと面クリアという、きわめてオーソドックスなタイプだ。と、これだけでは、いままで登場してきたシューティングゲームとなんら変わりはない。では、どこにこのゲームのウリがあるのか？

それは、スプライトの回転ルーチンを利用した多関節キャラなのである。各面ごとのボスや中ボスなどに、そのルーチンを利用していて、実に多彩な攻撃をしかけてくるのだ。たとえば1面では、メカニカルなカニのお化けが登場する。手や足の関節がわしゃわしゃと動き回りながら攻撃してくるのだが、この動きがとても自然で違和感がない。ほかのキャラクターでも効果的に使われていてずいぶん感心した。なるほど、これならウリにしたくなる。

オプション！ ◆◆◆◆◆◆◆◆◆◆◆◆◆

ソル・フィースでは、4種類のパワーアップユニットがある。それぞれ、自機の上下に装備でき（取ったときの状況による）、装備されたユニットは、水平位置から斜め40度近くまで、攻撃方向を変化させられるのが面白いところだ。ショットボタンを押

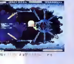


しながら前進す[illegible]り、撃出するとユ[illegible]ショットボタンが押される[illegible]位置で固定される。ここにも、スプライトの回転ルーチンを使っているので、微妙な角度で調整ができる。状況に応じて火力を前面に集中させたり分散させたりできるのは、なかなかいいアイデアだと思った。オプションのレーザーもきちんと斜めに飛んでくれて見ているだけで感心できる。

あと、この手のゲームにありがちなスピードアップのアイテムは、存在していない。自機のスピード設定はゲーム開始時のメニューにあるコンフィギュレーションモードで自分好みに選べるのである。これは、結構良心的。もっとも最高速にしてなければクリアは難しいかもしれないが……。欲をいえば、イメージファイトのようにゲーム中にも自分でスピードをコントロールできればもっとよかったんだけど。

バランスはどうなっている ◆◆◆◆◆◆

このゲーム、ノーコンティニューで全面クリアは難しいが、根性があれば普通の人でも大丈夫。難易度は少し高めだ。でも、楽することはない。敵の攻撃はそれなりに激しいが、一度くらい死んでも必ず復活チャンスがある。それに、一度死んだとしても、そのあとすぐブラスターの入ったパワーユニットが出てくるので、最低限のパワーアップ[illegible]できるのだ[illegible]



一見怪々[illegible]

各面の長さも適当だし、ボスキャラ到達までに適度に中ボスが現れたりして、プレイ中にだれてしまうこともない。それに、アイテムを温存させることが、最低限装備でクリアできる。僕がこのゲームをプレイして、いちばん評価しているのはこの部分だ。「一度死んだらそれまでよ」的な難はゲームは嫌いなのだ（ちなみに僕の好きなゲームは「ドラゴンスピリット」［illegible］ーダイン」。両方とも努力と根性さえあればノーマルショットでもクリアできるんだ）。

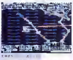
さて、ボスキャラ。これもよくできている。動きも凝っているし、攻撃パターンも多彩だ。また、ボスのつもりで根性入れて倒したキャラクターが実は中ボス!? と、びっくりすることもある。逆に、シューティングに慣れていない人には、多大なストレスを与えてしまうかもしれないけど。

そうそう、自機と地形との当たり判定は致命的ではなく、当たってしまうとただ単に跳ね返されるだけである。とはいえ、当たったときかなり予想外な方向に飛ばされてしまうので注意したほうがいいぞ。

面構成 ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆

[illegible]

まず、ゲームはソル・フィースがワープアウトしてきた宇宙空間から開始される。最初の面とあって難しいところはない。しっくりプレイしよう。[illegible]



文句なし[illegible]



ほら[illegible]





▲敵[illegible]
▶もうな、もいわん、気合いでよけろ

イン、はっきりいって弱い。攻撃を軽いフットワークでかわし、倒してしまおう。

2面は、敵の人型兵器工場の中を進んでいく。ここでは、ふんだんに多関節キャラが登場する。銃頭をつけうにと進んでくるしゃくとり虫のようなヤツ。腕をぶんぶん振り回すラジオ体操野郎、そしてメカニカルな触手。この触手は、当たり判定が先端部分だけ、というのを覚えておこう。ボスについては、ノーコメント。

兵器工場をあとにして、3面は敵が打ち上げた人工太陽の近くを通過していくことになる。背景は美しいが、敵の攻撃はかなり激しい。特に、ミサイルを乱射しながら迫りくる2機の中ボスがきつい。この面のボスキャラの出現にちょっとした演出があるが、一見すると何が起こったかわからない。まあ、気にするほどのことでもないが。

4面は、突如現れた巨大戦艦との対決である。少し、ありふれた感じもするがなかなかの迫力。敵の砲台が撃ってくるレーザーをよけるのがちょっと大変かな。

5面では、冥王星付近にある建設中の敵の基地を進んでいく。途中、後ろからやってくる巨大な作業用野郎が出現。うまく誘導して障害物に当てるか、バルカンファイアーを使う。できれば後ろに回って倒そう。ボスの場面はかなり集中力が必要。

そして、決戦前の6面。途中はともかく、この面のボスキャラは凄い。なんといっても「全ては南京たますだれ」なのである。なんのことかわからないって？ それはねぇ、敵のレーザー攻撃を見ればわかるよ。で、レーザー攻撃をよけるだけでも苦しいのに、いくつの攻撃パターンがまたうんとこさあるのだ。ここまでくるとかなりしつこい。もう、攻撃のしがいがあるなんてもんじゃない。それだけにようやく倒したときの感動は忘れられないけど。

いよいよ最終面の7面では、地球が舞台で最後のボスとの対決。[illegible]は自分の目で確かめよう。

全体の印象は？ ◆◆◆◆◆◆◆◆◆◆◆

総評的にこのゲームの評価として、合格点をあげられると思う。場面ごとのつながりもなかなかいいし、ボスキャラもいい味だしていて、プレイヤーを驚かそうという努力が見られる。

しかし、画面全体の印象はあまりよくないと思った。きちんと上下スクロールもするし、一部ではラスタースクロールまでもやっている。でも、やっぱりパソコンゲームだなあ、と感じてしまうのである。そう感じさせてしまうのはやっぱり描き込みが足りないせいだろう。オープニングのグラフィックと同じようにね。ゲーム全体のバランスも取れているほうだと思うし内容もいい。あとはグラフィックデザイン。ここさえ直してしまえば、かなりアーケードゲームのレベルまで迫ることができるだろう。期待大ですね。

やっぱりMIDIだなあ

例によって例のごとく、このゲームもMIDI（MT-32系）の音源に対応している。一応、音色はオリジナルの音色を使っているようである。ウルフ・チームが制作しているゲームのFM音源を使用している僕は、以前より同じ音色を使っているのは聞きノ…承知のとおり、そのため、別のゲームでも[illegible]れとい[illegible]てほかのゲームと音色って変わるようなことはない。ソル・フィースでもその事実は変わらないのである（いろいろと事情があるようだけれども、どうにかしてほしいものだね）。まあ、MIDIもっている人は1度聞いてみることを勧める。僕自身、試しに聞いてみたが音源が変わるとずいぶんと雰囲気が変わるものだと思った。ということで、僕作りのほうもさらに頑張ってくれるといいですね。

総合評価	0 — 5 — 10
グラフィック	★★★★★★
多関節	★★★★★★★★★
音楽	★★★★★★
効果音（PCM使用）	★★★★★★★★
ビジュアル	★★★★★★★★

銀河の歴史がまた1ページ

かねこ しゅんいち
金子 俊一

すでに発売されている「銀河英雄伝説」の続編。前作と比べてシステムもパワーアップされ、ありがたいことに原作を知らない人のための指令官データやオート機能もついている。もちろんMIDI対応だ。



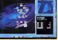
最近のシミュレーションゲームは力が入りすぎだ。デコレーションの部分ばかりがクローズアップされるようになり、マップの拡大化、ユニット数の増加、派手なビジュアルシーンといった具合に肥大化されてしまった。そう、シミュレーションは時間がかかって当たり前、24時間戦えますか的なゲームばかりになってしまったのだ。

それではいただけない。私はめんどくさがり屋なのである。中ダレしてしまうようなゲームではすぐに飽きてしまう。別にシミュレーションが嫌いなわけではない。その昔「サウザンティスの冒険」というOh!PCに載っていたゲームがやりたくて、H K氏の家に何泊もした私である。寝ている彼を横目に必死に斧を動かしていたものだ。

こんなのあってもいいんじゃない？◆◆

大架裟なものじゃなくていい。手頃なシミュレーションウォーゲームが欲しいのだ。ほんの短時間で終わるシナリオ、適度に小さいマップ、限られたユニット数。これらの条件を満たせばコンピュータの思考時間が大幅に減る。なによりも大切にしたいのは無駄な時間の消費がないことだろう。はっきりいって待たされるのはいやだ。

で、銀河英雄伝説である。アニメーションを使った戦闘画面なのに1～1時間もあれば終わるシナリオ。現状のシミュレーションウォーゲームとしては、まさに短期決戦[illegible]ゲームとしているのはその続編・銀河英雄伝説IIである。前作に比べパワーアップは著しい。グラフィックも強化され、まるで肥大化への道を歩んでいるかのように見える。確かにプレイ時間は長くなっているが（1シナリオ2～3時間）、まだバランスを崩しているレベルまではいっていない。救われた！



X68000用 5"2HD版 4枚組 8,800円［税別］
ボーステック ☎03（780）4711

原作を知らない人のために◆◆◆

銀英伝シリーズは田中芳樹の同名小説が原作となっている。小説について私は外伝しか読んだことはないが、三国志に近いものを感じた。吉川英治の三国志にハマったことがある人ならば十分ハマれるだろう。同名のアニメもあり、グラフィックや音楽関係はこちらのほうがマスターになっているようだ。ぜひご覧あれ。

銀英伝および銀英伝IIをプレイした感じでは、原作に対する知識は必要である。シナリオや登場人物の設定などは原作に忠実に作ってある。宇宙を舞台に銀河帝国と自由惑星同盟が戦っていることや、せめて登場人物の主だった面々がわかるぐらいでないと面白さは半減してしまう。目の前に黒色槍騎兵と呼ばれる艦隊が出現してもビビらないのは失礼にあたるというものだ。まあ、ゲーム中にコメントひとつで登場人物の経歴が見れるので、知らない人はこれを熟読しておくべし。ちなみに私は銀英伝を遊んでからアニメを見て本を買った人だ。

ウォーゲーム・銀英伝◆◆◆◆◆◆

一般的なシミュレーションウォーゲームと銀英伝IIとの違いを探ってみよう。ここでいう「一般的な」というのは光栄シリーズや大戦略シリーズのことである。

HEX戦ではない　現在地などは座標で表される。戦争とは違って同じ座標にいくつでも戦艦を置くことができる。だって宇宙だもん。Z軸だって無限である。

ターン方式が採用されているのは同じなのだが、**ターン終了と宣言するまでは何度でもやり直しがきく**。航路の変更をしたくなったら、そのまま新たに入力すればよい。最後に入力されたデータが選択される。X68000を買うときに、ACE HDにしようかALEにしようか悩んでるうちにSUPERが出てしまったような優柔不断な人間には最適のやり方である。

最大で32ユニットまで同時に扱える。しかし、毎ターンで32回も命令を出していたのでは時間と体力の浪費というもの。なぜなら**命令は全部のユニットに出さなくてもよい**からだ。このゲームには提督というものが存在している。1人の提督には大抵4～5ユニットが指揮下にあり、いっしょに行動している。これが1艦隊の単位である。つまり、提督ユニットに指示を出せば5～6ユニット同時に命令を出せるのだ。

でもって、**命令は毎ターン出さなくてもよい**。ユニットを移動させたかったら、そのユニットをクリック、メニューから移動コマンドを選んで、行っていきたい地点をクリックするだけだ。その移動には何ターン必要か教えてくれる。あとはその移動が終わるまで放っておけばいい。

さらに、**部下に作戦を伝えれば、作戦終了まで勝手にやってくれる**。たとえば「ここと、ここを通ってあの惑星を占領してくるように。あとはよきにはからえ」などと命令すると、そのとおりに実行してくれる。もちろん、途中での作戦変更や中止もできる。極端な話、1ターン目に作戦とそのコースだけ決めておけばあとは放っといてもカタがついてるはずだ。うん、らくちん。



3分割マップで艦隊の陣形を[illegible]

陣取りゲームなのね◆◆◆◆◆◆◆

このゲームには帝国軍と同盟軍しか存在しない。どちらかを選び、勝利条件に達してしまうが勝ちだ。非常に簡単なルールである。基本的に勝利条件とはマップ上にあるすべての惑星を占領すること、もしくは敵が全面撤退するまで叩き潰すことだ。そういう意味では、このゲームは見事に陣取りゲームである。しかし、ただの陣取りゲームでないところはシナリオ制が導入されている。しかも想像がつく、シナリオごとに異なった目的があり、これも勝利条件になるのだ。

帝国軍を選んだ場合、特定の惑星の占領が主な目的となり、同盟軍だと、その惑星を防衛することが勝利条件になっている。つまり、完全制圧ではなくても軍事的に重要な惑星だけを占領下とすれば、そのシナリオは勝利した、と認めてくれるわけだ。無駄な時間の浪費につながらない方法である。シナリオは5本、それぞれに帝国側、同盟側の勝利条件が決まっている。

で、これらを通してやるのがキャンペーンモード。キャンペーンモードでは特別ルールがあって、これまた遊び心をくすぐるものがある。そのルールとは、提督にレベルがあって、戦闘結果によってレベルが上がるのである。レベルが上がると先制攻撃をかけやすくなる。旗艦部隊の艦数が増え[illegible]







ゲームの途中で評価がわかる

そのほかにも「以前のシナリオで占領した惑星はその後のシナリオで登場する場合にちゃんと自軍のものになっている」とか「1度戦死した提督はそのキャンペーンには出られない」などがシナリオ単体でやるのとは異なってくる。

さらに最終シナリオで勝利すればエンディングも用意されているらしい。まさにいたれりつくせりである。

おりこうファジーじゃないけれど◆◆

嬉しいことにいろんなコマンドに自動という選択肢がついている。たとえば補充部で補給コマンドを実行した場合、本来プレイヤーは補給が必要な部隊をいちいち選ばなければいけないのだが、自動にしてしまえば部下が選んで補給をしてくれる。極めつけは「ゲームそのもの全自動」というやつである。アニメーションモードにしておけば立派な環境ソフトになっているわけだ。初心者だったら全自動モードを見ながら研究すればいい。きっと役に立つはずだ！

コンピュータのアルゴリズムは秀逸とはいいがたいが、十分に楽しめる。索敵する楽しみがあるというのだろうか。マップ自体はすべて表示されているのだが、敵のユニットは索敵をして見つけた敵のみが表示されるのである。気づいたときには囲まれていたなんてことも起こりうるのだ。

敵艦隊のコースがわからないまま目的の惑星に前進するのはなかなかに怖いものがある。たまたま遭遇した敵が不敗の魔術師ヤン・ウェンリーの艦隊だったら……。攻め込んでくる提督が常勝の天才ラインハルト・フォン・ミューゼルだったら……。考えただけでも鳥肌が立つ。このゲームにはそういう恐怖が常にある。その1歩先は闇[illegible]がこのシステムの最大の面白さになっているのだ。

ちなみにコンピュータのアルゴリズムに飽きた人向けではないのだろうが、銀英伝IIになって対戦モードも用意された。ポピュラスが流行ったあとなので困惑的な気もするが、X68000を2台も必要としないからよしとしよう。



ほしい戦艦が入る。艦の設計はうれしい

音楽性は？？？◆◆◆◆◆◆◆◆◆

最後に音楽について触れておこう。内蔵音源のほかにMIDI（MT-32）に対応している。HELPキーを押しながら立ち上げればMIDIポートからオープニング曲からFM音源が鳴る。もちろん、ゲーム中などコマンドの選択でFM音源かMIDIかを選択できる。MIDIボードがささっているとFM音源は鳴らなかった前作と比べてあわあわよって、システムとしては着実に進歩しているし、ユーザーの声もずいぶん反映されているようだ。これはとってもよいことだと思う。

しかし、肝心の音楽に関してはちょっといただけない。おそらくアニメの銀河英雄伝説のサントラ版が何曲からの選曲なのだろうが、耳慣れしてないこともあってイマイチである。クラシックを使っていた前作のほうが聴きごたえがあった。効果音においては合格点なので、BGMを選択しないで自分の好きな音楽をかけるのもいいだろう。お勧めはベートーベンの交響曲3番・英雄である。前作に使われていた曲なので違和感がまったくないのがよい。

趣味の総評

フルマウスオペレーションを可能としていてなかなかに気持ちがいい。艦隊をマウスひとつで運用してしまうあたりは、まさに「気分は銀河帝国元帥」といった感じである。そのマウスのシステムも、以前に増して磨きがかかってきている。左クリックが決定及び確認、右クリックが取り消しという事項のすべてにおいて貫き通してある。まるで蜃気の隅をつつきまわしたようなソフトで、いたるところに気配りが見られる。うん、偉いぞね。

総合評価	0　　5　　10
システム	★★★★★★★★
グラフィック	★★★★★★★
サウンド	★★★★
インタフェイス	★★★★★★★★
[illegible]	★★★★★★★★★★

皇帝陛下はマウスを司るのである



Komura Satoshi
古村 聡

このシミュレーションゲームがX68000に移植されると聞いて，忠犬ハチ公のように待っていた人もいることでしょう。いろいろな機種を経て，やっとX68000に，ということで実際にも期待できそうですね。

X68000にシュヴァルツシルトが移植されました。このゲームはプレイヤーがサンクリ星国皇帝となり，工場の建設，宇宙戦艦の建造などを行いつつ，サンクリ星国を戦争で偉大なものにしていき，最終的にはジロ星団宇宙全体を平和にするのが目的，というシミュレーションゲームです。先にPC-9801，PC-8801，MSX版が出ていて，しかもなかなかにつまいシナリオ運びで評判がよかったので，ご存じの方も多いことでしょう。X68000用もずいぶん前から発売が決定していたのですが，完成までには結構時間がかかったようです。

で，レビューをやるためにこのゲームを始めたのですが，実をいうと原稿を書く時間が惜しいほど，ハマってしまっているんですよ。仕事だというのを忘れて，シャカリキになって遊びまくっていたりするんです。

吾輩は陛下である

吾輩が大銀河シュヴァルツシルト（シロ）星団，サンクリ星国の皇帝である。皇帝陛下様と呼べ！　うう〜ん，気持ちがよい。ふふふ。私はこれから楽して暮らすのである。なんてったって皇帝陛下様なのである。

む，これ，そこで何を難しい顔をしておる？　あ，なになに，皇帝なんだから仕事が山のようにあるのか。うむ，わかったわかった，よきにはからえ。

えっと，まず何ができるのだ？　ふむふむ，画面右上のメニューにできることが書いてあるわけだな。うむうむ，艦隊の移動，演習，いま持っている宇宙戦艦隊に命令を下せるわけだな。あれ？　艦隊は4つしか持てないのか？　正規の艦隊が4つと2個の警備艦隊が合わせて6つまでなのか。ふむ，それでは，どうやって戦力アップを図るのだ？

ははっ，戦力アップを狙うためには2つの方向があるとな。ひとつは戦艦の数を増やすこと，そして，もうひとつは戦艦のモデルチェンジ，すなわち性能を上げること（当然，性能が上がるのは新しいモデルになってから建造された艦のみ）なわけだな。ただし，違う型の戦艦は同じ艦隊には所属できないのか。まあ，型が新しい艦を大量に配備すればいいわけだがな。そうすると金が足りなくなる？　難しいところだのう。

ところで，平和的な皇帝のために，戦争をしなくてすむ方法というのはないのか。えっと，メニューを見てと。おお，外交なんてものがあるではないか。クリック。

おや，「反乱勃発による政情不安のため諸外国との外交活動はまったく不可能です」だと。するとまずなにがなんでも反乱を武力制圧しなくてはいかんわけだな。よし，しかたがない。反乱軍をクリックして情報を見てみると，つっきゃーっ，反乱軍はとっちえのモデルの戦艦ではないか。おまけに反乱軍はどこかの星から援助を受けてるという話じゃないか。これなんで勝てるんかいな，まったく。

まあ，とりあえず戦艦のモデルチェンジをして，同時にいまのモデルの戦艦も増産してと。

よし，戦艦も揃ったところで，反乱軍鎮圧に出撃！

ふたを開ければ

吾輩が大銀河シュヴァルツシルト統一を成し遂げた偉大な皇帝なのである。皇帝陛下様と呼べ！　うーん，何回やっても気持ちがよいぞ。

いまや大銀河シュヴァルツシルトを統一してしまったのである。統一というとちょっと語弊があるかもしれないな。とりあえずもう反乱を起こすものがいなくなった，というのが正解かもしれないのだが，とにかく戦いに勝ったのである。いやいや天下太平めでたいめでたい。

実際やってみると，楽勝であった。戦闘モードでの戦いの勝敗は自分の星に攻撃をされてしまう前に敵の旗艦を倒してしまえば，こちの勝ち，というとっても簡単なルールなのであった。だから，よっぽど敵のほうが戦力的に強すぎるなんてとき以外は，敵の惑星なり旗艦（印がついているのですぐわかる）を4艦隊で一気に攻め込むという戦法でなんとかなるのだ。

敵が攻めてくるときには，同盟国を組んでいる国の軍隊を呼ぶというのもいい戦法だ。なにしろ同盟国軍の艦隊も，全艦隊がとにかく敵の旗艦に突っ込んでいくので，自分の艦隊は惑星の近くに待機するようにしておけば，そこへ敵艦が来るころには敵旗艦は相当消耗している。そこで敵旗艦を叩けば，結構楽に敵を倒すことができるのである。ちなみにこの戦法はオペレーションが面倒くなったときなどにもなかなか有効である。同盟国が勝手に敵をやっつけてくれるから。

どちらにしても戦闘のときはそんなに深く悩む必要はない。このゲームではとにかく「数の勝利」という言葉がぴったりくるのである。

反乱軍を鎮圧してしまうと，やっと他国との外交ができるようになる。同盟と不可侵条約に関しては積極的に行うようにすべきであろう。なにしろ，同盟軍は多ければ多いほど戦いの際にも有利だし，不可侵条約を結ぶことができるのなら，一度に相手にする国が減るわけだから有利になる。また，宣戦布告はそれほど自分から行う必要はない。すべての国がというわけではないが，ほとんどの場合，敵となる国は向こうから宣戦布告を行ってくるからなのである。

とにかく，このゲームでは戦闘そのものではなく，投資の振り分け方でだいたい勝敗が決まってくる。戦艦をどれだけ持っているかで戦闘の結果の8分は決定されるといっていい。とりあえずは，いまある型の戦艦を増やし，一段落したら新型の戦艦開発に投資して，それを順々に配備していく。これが銀河皇帝への道である。

時間がかかっただけのことは

皇帝になりきるのはこれくらいにしてと。感想なのでありますが，非常に面白いです。ハマります。まず，反乱軍を戦い終わると新たな敵が，そいつに勝つとさらに新たなる敵が……という具合に進んでいくので，なかなかやっていて楽しいです。うまくゲーム進行を考えてくれていると思います。そうそう，目の前の敵を倒すと次なる新たな敵が出てくる（当然，前の敵よりも強い）わけですから，目の前の敵に勝てそうだからといって気を抜いてはいけないのですよね。ちょっと多めに新しめの戦艦を作っておくと，あとの展開がとても楽，かつスピーディになります。これが銀河統一のコツです。

あ，統一で思い出した。このシュヴァルツシルトなのですが，実をいうとシュヴァルツシルト銀河に敵なし！　……というさっきのところでゲームが終わりではないのです。というのも，実は○○で○○な敵が攻めてきて……（あんまりいっちゃうとまずいだろうな）。とにかくここから先が最大のヤマ場であり，真のシュヴァルツシルトでもあるのです。

このゲーム，システム的にも大変よくできています。操作はオールマウスオペレーションでスイスイいけちゃうし，コマンドも比較的種類が少ないのでシミュレーションゲーム特有のかったるさはまったくありません。

また，グラフィック画面もX68000用に描き直されていてきれいになっています。外交や宣戦布告されたときには相手の国の元首の顔が出てきたり，戦艦ができたときなどには戦艦の絵が出てきます。戦闘時には，たとえば惑星を攻めるときには艦隊がバシュバシュバシュッとミサイルを打ち込む場面が，しっかりとアニメーションしてくれます。このアニメーションのときにディスクを読みにいくんですが，読み込むのが速くて気にならない程度になっています。X68000の性能を生かした移植となっていて，システム的にも大変よくできている，と私は思います。はい。



宣戦布告したが，返り討ち……



一難去って，また一難



宇宙要塞というのも



戦闘場面はアニメーションで見せてくれる

一般ピープルおすすめゲームだ

シミュレーションゲームというと面倒臭いゲームの目的がわかりにくい，ゲームが単調になりやすい，ということが問題とされてます。これはいままでシミュレーションゲームではある意味避けられない問題でした。シミュレーションゲームの特徴が，シミュレーション＝現実の模倣にある以上，現実の世界にある雑多な事柄を引きずってしまうのはどうしてもやむをえないことですからね。

このシュヴァルツシルトはどこかの星々という架空の世界のシミュレーションです。つまり人が作り上げたフィクションの世界の話であって，現実をシミュレートしたものではありません。このゲームではそれがとてもよい方向に作用しています。ゲームの先が読みやすくなり，ひとつの目標が達成されたら，また次の……とうまくプレイヤーをゲームに引き込んでいくことに成功しています。

シミュレーションマニアの人であればゲームを行うこと自体をゲームの目的と捉えることもできるのでしょうが，普通の人の場合はそうもいかず，ゲームを進めるのになにか目標がないとやっていけないという人が多いと思います。

そんなわけで，このゲームはシミュレーションマニア以外の人にも絶対的にお勧めできる1本です。特に，シミュレーションは面倒臭いと避けていた人には，シミュレーションゲームへの入り口として，ぜひともやってもらいたいと思います。

独断評価	0 ─ 5 ─ 10
とっつきやすさ	★★★★★★★★★★
最初気分	★★★★★★★★
おまけ度	★★★★★★★★★
オススメ度	★★★★★★★★



X68000用　5"2HD版2枚組　12,800円（税別）
工画堂スタジオ　☎03(363)7724



わっ，戦艦ができたぞ



情報はウィンドウ上に表示される

クリスマスを暗いダンジョンで祝おう

七面鳥の代わりに丸虫の肉で

Ogikubo Kei
荻窪 圭

いよいよあのダンジョン・マスターの続編が一段と厳しくなって登場。"続"というだけあって，もちろん前作のキャラクターデータを使うこともできる。フルアニメーションのオープニングも見ものだ。



前作のダンジョン・マスターを遊んでない方，本作品は前作を知らなくてもまったく問題なく遊べます。本当に大丈夫です。しかし，シューティングゲームを知らない人がいきなりナイアスやイメージファイトを始めてしまうようなものです。魔法を唱えるのにあたふたし，戦いに手こずり，ダンジョンの要所を見逃し，無数の傷を負うでしょう。ダンジョン・マスターを終わらせる必要はまったくありませんが，せめて地下10階くらいまでは経験しているとずいぶん違います。ローヒ，カオスを無事融合させた人でも，私のように1年もブランクがあるとなかなか勘が取り戻せないので，しばらくダンジョン・マスターのいちばん成長したキャラクターディスクで遊んでみるのもいいでしょう。その間，ちょとばかりレベルをあげておくとよりGoodです。

●　●　●

本当はいやだったのだ。もう2度とあんな目はごめんだったのだ。食料さえ豊富でない，ちょっと気を抜けば時鐘が襲い，曲がり角には怪物が息を潜め，傷を負い，いつ果てるとも知れぬ命といつ果てるとも知れぬ迷宮の持久走。

しかし，穴があれば入るのが男，古今東西そういうものなのである。穴があったら入れたいじゃなかった，入りたいのだ。たとえ，どんな怪物が潜むやもしれぬ……であっても！



X68000用　5"2HD版2枚組　8,800円(税別)
ビクター音楽産業　☎03(423)7901

旅の道連れを決めること ◆◆◆◆◆◆

カオスの逆襲。カオスってのは混沌のこと。カオスが悪者ってのはエントロピーの法則を信じている私には少々気に入らないが，エントロピーが増大すること自体が悪なのかもしれない。私の部屋がすぐ汚くなってほこりが溜まるのはエントロピーの法則が働いたからにほかならないのだが，女の子はそれを見て"悪"だという。そんなものさ，私が悪いわけではないのだ。

逆襲してくるならこちらも返り討ちにしてくれよう，というわけで，旅の道連れを決める。どうするかは私の勝手。

選択肢は2つだ。新たに閉じ込められた勇者(今度は勇者ではなく，チャンピオンだ)を4人選ぶか，前回の冒険で道連れにした者たちの手を再び借りるか。

今までの常識からいって，前作のキャラクターを持ってきたほうが楽し進めるはずだ。しかし，である。そうはいかないところがダンジョン・マスターだな。それとも，私があまりにも成長させないまま終わらせてしまったからかもしれない。とにかく弱い，弱い。まあ，普通の人は前作のキャラクターを"MASTER"くらいにはしてるだろうから，大丈夫だろう。チャンピオンたちはみなEXPERTくらいだ。その代わり，ヘルスやスタミナが大きい。どっちもどっちだ。今回の新キャラクターは人間でないやつらが多いので，人間パーティを組みたいときはなおさら前作のキャラが欲しいだろう。今回登場したケンタウロスを使いたいという気もするが，私は一応義理と人情で前作のヒィッッッサとゴスモグとウーフとダルー――を使ってやるのであった。我ながらいやだろうが，旅は道連れ世は情けってもんだ。

ちょとばかし，前作とシステムが違って，道連れを4人選ぶ牢獄と旅するダンジョンが分かれている。牢獄で4人選んでセーブするか(今回はなかなかエグいやつらばか

りいるぞ)もしくは前作のデータディスクを使用し，"冒険の準備"をするのがよいのだ。

何はともあれ準備が整ってわいたところで，冒険の前にお買い物もしたかったけど，させてくれなかった。今回は丸裸の状態でダンジョン突入だ。

暗闇で丸虫に囲まれること ◆◆◆◆◆

突入。暗闇。ここはどこ，何者かに襲われている。痛い。足元にたいまつ。拾って灯す。鉄格子。振り向く。食べられるが味はOOP！　な丸虫のアーマードバージョン！　痛痛痛。

はっはっのていというやつ。では後世の勇者たちにメッセージを残そう。いきなりアーマード丸虫の下敷きに出現れるので，ぬかちとりあえず逃げるべし。アーマード丸虫は3匹しかいない。ただし，鉄格子の穴はアーマード丸虫出現フットスイッチになっているので，格子を開ける鍵を見つけるまでは決して鉄格子には近づかぬよう。

武器，とりあえず逃げ，明かりを灯し，アーマード丸虫をやっつけ，やっつけ，やっつけようと思ったのはいいが，戦士は丸腰，魔法使いは……魔法を思い出せない。前作の魔法をそのまま使えるのに，魔法をメモした紙をなくしてしまったのだ。ファイアーボールの魔法はどこだ。しかたがないのでダンジョン・マスターガイドブックを……。



魔法を駆使し，3匹のアーマード丸虫を倒し，肉を腹へしまい，部屋にあるアイテムをかき集め，よく眠り，体調を整え，鉄格子OPEN。

ここからが冒険の始まり。

無名戦士の骨のこと ◆◆◆◆◆◆◆◆

前作ダンジョン・マスターのおかげでマウス・トラックボールの左ボタンがいかれ，以来，両脇についているサブボタンを使っている。マウスを見ると，わしづかみする癖がついてしまった。

旧冒険したダンジョンは下へ向かって進む縦突猛進だった。そのフロアを制覇したら下へ，下へ，深く，深く。

CHAOS STRIKES BACK(邦題：カオスの逆襲)はそんなあまい場所ではない。いきなり地下3階である。下へいけばいいのか上へいけばいいのかわからず，目に見えない落とし穴はあり，目に見えないワープゾーンはあり，いきなりレッドドラゴンはいる。ないのは愛と希望だけだ。

最初のトラップは空フラスコ。空フラスコが欲しい。ないと，怪我も直せない。取りにいくと，必ず穴に落ちねばならない。どうするか。答え，フラスコはきっとほかにもあるさ。

魔法の地図。これが最高の秘製かもしれない。右手に持つと，7ブロック分の地図が見られる。これを方眼紙にでも写し取ればいいのだが，いかんせん，マッピング習慣のない悲しさ。つい地図に頼ってしまい，根気や目印識能がおろそかになる。

今回のダンジョンは恐怖だ。あまりにも先は長く，あまりにも複雑で，あまりにも奇怪。マッピングなしでどこまで行くかが挑戦。ひとつのフロアがいくつにも分割され，そこそれぞれに階段があり，壁には壁のようで壁でない壁がたくさんあるのだ。

地図表示の上に，呪文をのせる。壁でない壁や見えない落とし穴(冷か撒らしていれば魔法を使わなくてもわかるが)が見える。ダンジョンにいるモンスターの場所がわかる。そういう呪文がオリジナルのAtari ST版に加えられている。本来本元に手を加えるのは望むところではないのだが許そう。それほどカオスの逆襲は過酷。

歩いていると，ひとつ目の祭壇に出た。ワープ装置が設置されており，四方，選んだほうへワープする。どちらにも復活の祭壇があり，骸骨にアイテムがひとつずつ置いてある。これを選ぶかでとりあえずの将来が決まる。

運の良し悪しはない。強いていえば，剛





の剣は呪われていることくらいだ。この祭壇選択の場には再びふたたびおとずれることになるだろう。

ひとつを選んでさらに地下へと降りる。こん棒を持ったやつらを蹴ちらし，さらに下へ降りる。広い部屋だ。そろそろ地下8階くらいだろう。骨が落ちている。無名戦士の骨だ。思ったとおり，レッドドラゴンがいた。やつがのろまなのをいいことに足元をすりぬける。レッドドラゴンと戦う気はない。

壁でない壁を魔法で見つけ，通り抜けるとワープゾーン。そこへとびこむと，再び祭壇選択の場。

道は無数だがひとつもないこと ◆◆

はあ，はあ，み，みずぅ。はあ，はあ。く，くいもの……腹が，食い物の元，スクリーマーの宝庫があったはずだ。あれはどこだっけ？　あ，思い出せない。み，みずぁ，水だ。前はなかったのに，いつの間に水が？　舌が浮いているけど文句はいえない。

あ，閉じ込められた。供物がある。斧棒を析れ。祈る。祈る。

明日はどこにある。

紹介するのこと ◆◆◆◆◆◆◆◆◆◆

というわけで，カオスの逆襲はいきなり神秘メニューから始まる。ひとつがフルアニメーションのオープニング。使って冒険の準備，ここではダンジョンに突入させるキャラクターのディスクをロードし，冒険の準備をする。その際，名前や容姿をエディットすることができる。そういうことらしい。だから，魔法使いサリーとか僧侶の一休さんとか戦士のバトロイド・バルキリーとか名探偵浅見光彦なんてのも作れるわけだ。

3つめがGO TO ダンジョン。例によって入り口の扉が現れるわけだが，新しく始めるときは道連れの選択になる。道連れを選択したらセーブして終了して冒険の準備をしてもう一度ダンジョン突入をする必要があるのだな。

今回新しく付け加わったのは，ところどころのBGMと，新しい魔法と効果音の充実(ときのこえはちゃんとあげるし，壁を叩くと音がするようになった)だけ。それから24KHzディスプレイ対応になった。24KHz対応ディスプレイを使っている人は，24KHz(つまり，PC-9801やFM TOWNSと同じような画面)でも遊べるようになったのも変更点だ。

あとは前作とまったく一緒。あ，BGMがついたと聞いてラッキーって思っただろ。ラッキーじゃないよまったく。ダンジョン・マスターは恐怖の雰囲気のゲームなんだから無粋なことはしないでもらいたい。静寂の緊張感が欲しいのだ。

ダンジョン・マスターは逆襲してきた。以前より複雑さを重視したマップとともに，トラップの数とプレイヤー心理を逆なでするいやらしさは天下一品というものだろう。

カオスの逆襲をひとことでいうと

かつて私は，ダンジョン・マスターを称して，アメリカの冒険文化がゲームになったものと評した。それは今回も貫かれている。ただし，前作のような教育的配慮がまったくない。いきなりバカ強いやつと遭ったり，その先で弱い食用モンスターと遭ったりという調子だ。道は常に選択されるものであり，他人の意見は参考にならない。丸腰で始めるため，前作当時よりずっと軽い頭けない装備でレッドドラゴンと出会ったりする。BGMはイトあるとオンメモリで動くことや，ディスク1枚であれだけ遊べるという優秀の売りは健在だ。

総合評価	0　5　8
スリル	★★★★★★★★★
サバイバル	★★★★★★★★★★
教育的配慮	★★★
不親切さ	★★★★★★★★★
手取り始めた	
あとの地獄	★★★★★★★★★

みんな知ってる定番野球ゲーム



簡単な操作法で野球の楽しみが味わえると、大人気の「ワールドスタジアム」。その"ワースタ"がついにX68000にも移植されました。これでストーブリーグも友達を集めて野球大会でもやれば、さみしくないかな？

Kageyama Hiroaki
影山 裕昭

今年は日本シリーズまあ…という間に終わっちゃってですね。ああ、つまらない。僕は巨人ファンですから。プロ野球シーズンも終わってしまいましたが、野球ゲームのシーズンはこれからです。X68000にもやっとワールドスタジアムが発売されました。ワールドスタジアム、通称"ワースタ"はファミコンゲームの不朽の名作"ファミリースタジアム"のお兄さんとでもいうべきゲームで、野球ゲームの決定版です。

いない選手は作れ

ワースタは12球団が登録されていて、アー…リーグ、カントリーリーグという呼び名になっています。球団名や選手名は実在のものをもじったもので、これがなかなか笑えてしまうのです。ジャイアンツの"まきばり"でウケてしまったのは私だけ？ 1球団は投手が5人、レギュラーの野手が8人、そして代打が5人いるので合計18人です。どの球団も代打になつかしの名選手がいて、それを使えばどのチームでも割と個性なく戦うことができます。

ところで、レッカルーズ（成績のこと）を選んでも、投手陣の中に野茂の名前がありません。野茂のいないバッカルーズなんて彼女のいないイブの夜のようなもの。ちょっと寂しいですよね。よく選手を見てみると、なんとなくデータがらしようです。阪神にはあの恐ろしいバースがいるし……選手データを変更できるエディットモードがあるからご安心召され。細かいところまで変更できるから、どんな選手も簡単に作れます。選手名、利き腕、バッティングフォーム、肌の色、バットの種類、ホームラン数。以上が野手・投手共通に変更できるもの。これに加えて、野手は打率、打撃力、足の速さ。投手はピッチングフォーム、防御率、スタミナ、球速、変化球の曲がり具合を変更できます。

これだけたくさん変更できると十分といえるでしょう。こだわる人なら選手名鑑やスポーツ新聞などを片手に、各球団のデータをそっくり打ち込むのも面白そうです。

簡単とはいったものの、数が多いから全選手のデータの変更にはそれなりの時間と労力を必要とします。その点、ちゃんとデータの保存ができるようになっているのはありがたいことです（当たり前か）。

球場は後楽園、甲子園、メジャーの中から自分の好きなものを選択できます。球場のグラフィックはどれもきれいで、後楽園や甲子園は本物をもじった広告などがあって雰囲気が出ています。メジャーは本当に広いし、イニングの合間には飛行船が飛んでいたりして、アメリカンな感じです。

さて、操作方法はといえば、ファミスタと同じです。しかし、ジョイパッドやジョイスティックによってはボタンがファミコンなどと逆になっているものがあります。これはファミコンなどでやり込んでいる者にはちょっとつらい。ソフトウェアでのボタンの逆転がほしいところです。

ルールは実際の野球と違うところがあります。ランナーのいない時に牽制球を投げてもボークにはなりません。また、10点差がつくとコールドゲームになります。

ひとりのときは

スポーツゲームは友達と対戦して遊ぶのが角打ち…それでも、ひとりで遊ばなくてはならないときもあるから、コンピュータの強さも気になるところです。そこでコンピュータの思考ルーチンを分析してみました。……はまあまあっぽい……。しかし……守備が問題。フライを捕るのはうまいけど、ごくたまにライナー性の打球が飛ぶと、頭の上を抜かれてしまうことがあります。決して捕れない打球ではないのに。それとゴロの処理ははっきりいってヘタ。横から回り込みながら捕るということをしないので、守備範囲が狭い。コンピュータだから守備は完璧だと考えてしまうのは、まったくの誤解です。そのうえ、相変わらず（ファミスタのときから）捕球したあとの送球がまぬけです。

「ピッチャー、振りかぶって、投げました。カキーン、打った。これは大きい。右中間まっぷたつ。ランナーは1塁を回って2塁へ。センターがやっと追いついて、セカンドへ返球しました。おーっと。これを見た打者走者はすかさず3塁へ進塁、ゆうゆうセーフ」

この場面では、打者走者が2塁ベースを踏む直前にセンターが捕球しました。ここでプログラムは返球先を決めるのに、打者走者の位置を調べます。すると走者は2塁ベースを踏む直前、つまり1塁と2塁の間にいることを確認したので、2塁へ返球したのです。これが人間だったらどうでしょう。いまから2塁へ返球しても間に合わない。しょうがないから、3塁へ返球しておこう、となるはずだ。ランナーがある塁に……返球する



X68000用 5"2HD 2枚組 9,800円（税別）
SPS ☎0845(46)5777



選手の……

ようにプログラムすることも簡単なはずなのに。全体的にコンピュータ側は先を読んだプレイができないのです。ワースタは人でワイワイとはしゃいで遊ぶのが面白いゲームだから、コンピュータの思考ルーチンはどうでもいいという感はありますがね。

なんといっても対戦

というわけで、コンピュータと対戦して遊べるのは、操作方法に慣れていない最初のうちだけ。やっぱり友達を引っ張ってきて対戦するのが正しいワースタの遊び方でしょう。

人間が相手となると気合いの入れ方もがぜん変わってきます。やっぱり、やるからには勝ちたいじゃん。ここでちょっと伝統的な必勝法を紹介してみましょう。

まずは、速い返球。これはレバーを倒すのと同時にAボタンを押せばいい。きわどいタイミングの内野ゴロや、外野からバックホームするときに使うと有効な技です。

それから2塁へ盗塁した走者を刺すとき、普通なら↑+Aですが、↓+Aを押します。↓+Aはバックホームですが、捕手は投手もしくは捕手の動きはホームベースの後ろになっているので、誰もいないホームに投げてしまうのです。するとあれっ、となる。投げた球はそのままゴロゴロと2塁ベースまで転がっていくわけ（球が高く浮かないから速い）。投手が捕ってしまわないようにしないといけませんが……。









試合の結果はナムコスポーツで

……もコツがあります。難しいことはなく、ストライクを投げるときはコーナーいっぱいを狙って投げる。ときたまボール球やフォークを投げるとさらに効果的です。これだけのことで、バットに当たってもなかなかヒットが出なくなります。

守備も最初はフライを捕るのが難しいけど、ボールの影の移動速度と、キュイイイインという音から高度を推定して、すばやく予想落下地点へ選手を移動させます。捕れないと思ったら前に出るのを止め、すこし後ろに下がって待ち、落下して1バウンド以上してから捕るようにしましょう。バッティングはなんでもかんでも引っ張るよりは、流し打ちしたほうがヒットが出やすいようです。

と、これだけ頭に叩き込んだら、最初はコンピュータと対戦してみましょう。ファミスタをやったことのない人なら、おそらく苦戦することでしょう。

試合開始時には「プレイボール」の掛け声がかかる。BGMの流れる中、「ストライク」「ボール」「アウト」「セーフ」とサンプリングされた女の子の声がスピーカーから聞こえてきます。審判は女の子のようです（スコアボードの審判の名前でもわかる）。なんともナムコらしい発想。ほかにもナッドボールなど「いてっ」「このやろー」、ホームランを打つと「あった、入った、ホームラーン」という声が流れます。

いっぽう、ホームランを打ったときの演出にも気持ちいいのだ。……歓声が流れて、画面にはデカデカとHOMERUNと表示され、それが……のように輝いて、おまけに飛距離まで教えてくれる。逆転ホームランだったりすると、ついつい今まで泣いていた方も抜けて……として使いへってしまいますよね。

チェンジのときには、スコアボードが表示されます。時計もちゃんと付いているから試合時間（現在時刻とは違うそう）もわかる。

そして、試合が終了すると、ナムコ新聞が今日の試合結果を伝えてくれる。一面に書かれる3行の文章が読んでいて面白い。スポーツ新聞のイタリアンな……がないか惜しい。

またまた要2Mバイト

うすうすね……と思いますが、このゲームもワールドコートに引き続き、要2Mバイトです。たぶんサンプリングデータがメモリを占有するのでしょう。サンプリングの数を減らしても（なくても）ワースタが遊びたいという人はたくさんいると思います。

SPSさんもワールドコートのときに同じような意見を聞いていると思うのですが、1Mバイトでも遊べるようにはできなかったのでしょうか。2Mバイトあるときだけサンプリング音などが流れるなどをやってくれればいいと思うのですが。遊べるゲームだけになんとかしてほしかったところです。

やっぱり、2人で

ワールドスタジアムは元祖のファミスタがファミコン用だったということもあって、非常に扱いやすく、それでいて奥の深いゲームです。ゲームの中身はそういうことでよくできていますし、声が出る、球場のグラフィックもよくできている、動きもいいなどと、見栄えもよくなっています。観衆はちゃんと人間の形をしているし、ホームランを打てば手を上げてよろこんで（騒って）くれます。移植はほぼ完璧といっていいでしょう。ファミコンやPCエンジンでやっても楽しめるゲームなので「わざわざX68000でやる」こともないと思う人もいることでしょう。しかし、パソコン版の特長としてエディットモードがあるので、また違った楽しみが味わえるのではないでしょうか。

総合評価	0 — 5 — 10
操作性	★★★★★★★
選手のキャラクタ	★★★★★★★★
グラフィック	★★★★★★★
おもしろさ	★★★★★★★★★
熱中度（1人のとき）	★★★★
熱中度（2人のとき）	★★★★★★★★★★

ああ,懐かしのフェアリーランド



アクションRPGの元祖ともいうべき,あの"ハイドライド"シリーズがX68000に初登場です。1作目と2作目は移植されていないので,初めて耳にする人もいるかもしれませんが,知っている人は本当に懐かしいですよね。

Takenaori Takahashi
高橋 哲史

ハイドライド3がついにX68000にも上陸です。いやあ,懐かしい。実はもっと早くに出ていたんですよね。これ,その当時はX1Dユーザーだったのですが,このへんが時代を感じさせる。このソフトも「X1turbo専用」となってきていて,愛機X1Dと泣き寝入りしていました。そこに「ノーマルX1でも動く」ということで,感涙して飛びついたのをいまでも覚えています。

さて,知らない人も結構いるかもしれませんので,ちょこっと"ハイドライド"についてご説明を。アクションRPG(以下,ARPGと略)の紹介などで「このゲームはハイドライドタイプのARPGで……」と引き合いに出されるように,この"ハイドライド"シリーズはARPGの元祖といえるものです。出た当時は「ドラゴンスレイヤー」「ハイドライド」ってなもんでおおいに話題になりました。"ハイドライド"は,まさに多くのRPGのひとつの原型となっている名作中の名作ゲームなのです。

また,今回移植されたのはPC-9801用に出ていたハイドライド3SV(スペシャルバージョン)ということで,X1版をやった人も楽しめるってです。

勇者でも最初は弱い ◆◆◆◆◆◆◆◆

さて,ハイドライド3も作目……まず……自分のキャラ……作目は……



X68000用 5"2HD版2枚組 7,200円(税込)
ブラザー工業(TAKERU) 052(824)2493

……職業は戦士,僧侶,……盗賊の4種類。温和で優しい性格の高橋くんは戦士を選んで勇者「りゅう」君を作ります。キャラ作成画面にいっぱいのパラメータが表示されました。……体力に腕力,魔力……

高橋「まあ……んならんでいる……」

りゅう「あーあ,ここでじっくり選ばないと,あとで苦労するかも。なんか,冒険の始まりを感じるなあ。実際に戦うのは僕なんだからね」

そんな心配をよそに,まず街に入って装備を揃えます。当然のことながらあまり高いものは買えません。武器はみんな貧乏なのです。地道に「食人樹」「ノエリー」を倒して,お金と経験値を貯めていきましょう。

高橋「おーい,てめえら金よこせー……ブッルー」

りゅう「ああー,見た目は似ているけど「樹木の精」さんとか,スライムくんとまよい人だから倒しちゃいけないのに……勝手に体が動くー…」

真の勇者は戦いの最中でも理性を失ってはいけません。ハイドライドには「善良なモンスター」と「悪いモンスター」がいるので,注意しなくてはいけないのです。間違って,善良なモンスター(近づいても攻撃してこない)を倒してしまうとパラメータのうちの「精神力」がぐっと下がってしまいます。そうなると,お店の人たちが品物をべらぼーに高く売りつけてくれるので,とても悲しい思いをし……ないのです。

編集「……勇者はいつも孤独なの」

りゅう「なんか違うんじゃない?」

さあ,ほどよくお金も貯まりました。思う存分お買い物です。ええと,武器,鎧に体力回復剤……,あら,食料もないわね。……

編集「あれ,お店から出たとたん歩も動けない……」

そっか,魔王の陰謀だな。「ヘッヘッ,出てきて勝負しやがれい,ビュンビュン」と剣を振り回す!

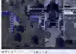
りゅう「い……荷物が重いんだよー…」

ちょーっと!,そうです,このゲームではアイテムの「重さ」が考慮されていて,あまり多くの荷物を持ちすぎるとその「重さ」のために動けなくなってしまうのです。この制限はある意味ではうっとうしいものですが,納得いく設定だし現実味が味わえていいと思います。

高橋「ちぇ,力のない奴だなー。しょうがない,食料捨てて,鎧も捨てて……ああもったいない」

りゅう「勝手なことを言ってくれるなあ。それなら最初のキャラクタ設定で,持てる重量というパラメータを大きくしてくれればよかったのに」





ハーベルの塔に ◆◆◆◆◆◆◆◆◆◆

ハーベルの塔というと,某国の「……ト」を思い出してしまうのは私だけでしょうか? 何はともあれ,いよいよ広まりつつあるのか登場している「ハーベルの塔」。突入です。この塔は前作の要ともいえる重要なポイントですので,心して挑みましょう。

高橋「200階もあるって,面倒臭いなー」

りゅう「僕なんか実際に歩いたんだよ。……ちは指を動かすだけなんだから……しっかりやる気出してよ」

200階建てというのは間違いないのですが,すべての階を回らなきゃ解けません」ってなことにはなってないので安心してください。肝心なのは,階のスイッチと,……階と最上階の……のはずですね。あ,あとこの塔にはエレベーターがあって,抜け道となるなんてこともあったりします。たまに,そこらにモンスターがいきなり通り抜けて登場してくれるのでびびってしまいますが。

編集「だあーっ。ボスだー……の一。の一,死ねー」

りゅう「うわーん,そんな無謀な戦い方しないでよ,痛いよー」

幾度となく非業の死を遂げながらも(誰のせいで?),冒険は徐々に進展していくのでありました。

天望の街 ◆◆◆◆◆◆◆◆◆◆◆◆◆◆

高橋「す……」



の塔はエレベータつき



……もの使い



地下の街

さっ!」

りゅう「……」

天空の街にご到着です。……どうやって来たかって? それは自分で考えましょう。案外簡単に来られると思うんですけど。あ,……ボスを倒さないと来られませんよ。まあ,これ以上はノーコメント。

ここでは高度な呪文を修得させる「魔術師の館」があったり,……を……してくれる……のいるお城があったりとなかなかいいところです。それと,実は「水の城」へはここから行ったりするんですねー。これは注意しないとよくわからないと思いますが,注意していればすぐわかります。……もないし,あと,肝心なのは思い切りと勇気です。

編集「わかった! ……っすればいいんだ。えいっ!」

りゅう「きゃあああっっ」

りゅう君がさまざまな試練を乗り越えてくれるおかげで数々の新たな明日がひらけるのです。ありがとう! 勇者りゅう君!

りゅう「もう身も心もボロボロ……」

水の城……と見せかけて地下の街 ◆◆

ふつうのきれいな人魚に会い……しぜんな……な地下の街です。

編集「だって水の城に行っ!……ドラゴンの角だか牙だかをほしい,とか……さいよ。こまぬかしやがるんだもん」

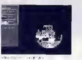
りゅう「そんなにうらいでたい……早く抜けられるんじゃないの」

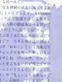
ここにドラゴンがいるらしいと聞きつけた我々はともかく地下の街への潜入に成功したのです。しかし,いかにも怪しげな街の空家には門番が立っていて,とても中には入れそうにありません。

編集「うーん,困ったなあ」

りゅう「君は頭を使うのが役目なんだから,ちゃんと考えてよ」

まあ,中になんとか入れるでしょうが,空家の奥にある洞窟に入るときはちゃんとランプとオイルを買っておきましょう。





……見る……

……なると地獄を見ます。私は一度見ました。

まだまだ冒険は続くけど… ◆◆◆◆◆◆

ドラゴンを倒したあともまだまだ冒険は続きますが,このへんで。

さて,気になる移植の出来の……ですがおおむね「良」といえそうです。X68000……のグラフィックは……(その中ら辺の人物の輪郭がなっと……ましたけど!) その時に出てくる……フォーフォント……には……での絵えがい……がりは……ですが,ちりを……でおって思わず「おお!」という……画面構成……ウィンドウを使って……しないですしね。ゲーム中,キャラの動きがちょっとぎこちないのが残念ですけど,昔のゲームということで……「経験値稼ぎ」や「ちょっと……なんて……ない……この……」が……ています。それも「ああ,懐かしい」……と感じて,かえって好意的に受け止められます(少なくとも私はそう……)。

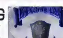
「ハイドライド」……知ってい……という……X68000……の人……もちろん,すでにX1などでやった人も,昔を懐かしんでもう一度プレイするのもオツなものだと思います。今回はTAKERUで販売ということで,比較的お手ごろな価格ですしね。

音楽も忘れちゃいけないぞ

ハイドライドは音楽も結構いけるのです。サ……ングなど……とはしてないし,ちょうとイースなどの陰に隠れてしまったこともあって,あまり取り上げられたことはありませんが,……曲が多いと思うのです。僕はこれのアレンジ版CDをいまでもよく聴いてますし。……お金に余裕がある方は,こちらもどうぞ。

総合評価	1 … 5 … 8
アクション要素	★★★★★★★★
RPG要素	★★★★★★★
移植の出来	★★★★★★★★★
操作性	★★★★★★
8ビット時代の名残	★★★★★★★★

万人向けの，手軽で高品質なRPG



中世ヨーロッパを舞台にしたRPG，ショートシナリオタイプなので飽きずにプレイできる。この作品にはシナリオ5本が用意されているが，別にシナリオ集も発売されているので，終わった方はそちらもどうぞ。

Urakawa Hiroyuki
浦川 博之

よく考えれば，しごくへのフィールド型のRPGの登場である。まだPC-8801が現役のころは，5×5マスしか表示されないマップの中を歩きまわって，ところかまわず戦って，いちばん強い奴を倒せばエンディングという"RPG"があっちこっちの雑誌に載っていた。いま，X68000からパソコンに入ってきた人には，この画面でアクション要素がないのが奇異に映るんだろうなあ。時代も変わったもんだ。まあ

おっと，思わず感傷に浸ってしまった。このブルトン・レイは，古き良き時代のRPGのスタイルを思い起こさせながら，実はそのシステムの品質は時代の最先端をいく。「自分が動かなければ敵も動かない」普通のRPGである（アクティブRPGではない）ぞ。念のため

ゲームのシステムは

ゲームの進行に関しては従来のRPGとなんら変わるところはない。オープニングでストーリー説明があり，あとはそれをふまえて人の話を聞けば，やるべき事と会うべき人がわかるようになっている。武器と道具をそろえたら町の外へ旅立つというお馴染みのパターンだ。

最近は，話の展開をスムースにするためにマップを一本道にして次の目的地がすぐにわかるようにするのが流行だが，ブルトン・レイは自分で目的地を探し当てなければならない。マップのサイズそのものはそう大きくないが，プレイしていると非常に広く感じる。

システムは比較的オーソドックスだが，そのなかで特徴を挙げるなら，徹底した自動化である。シナリオによっては2人か3人でパーティを組むことになるのだが，主人公以外のキャラクターの行動はコンピュータが担当し，プレイヤーはほとんど手が出せない。わずかに神官が使う魔法の種類と使う順序を設定できるだけである。メンバーに細かい指示をするわずらわしさを感じることなく，ひとりで冒険しているときも，パーティを組んでいるときも同じ感覚で遊べるようになっている。仲間のキャラクターは多少向こう見ずなところはあるが，しっかり自分が相手する敵を決めてかかっていく。バカな思考ルーチンにありがちな，ひとりで十分な敵に全員が飛びかかっていくといったことはない。

さらに自分の行動も，隣の画面への移動と，敵との戦闘についてはコンピュータに任せることもできるのだ。移動の際は複雑な地形でもきれいにまわりこんで歩いていくし，思いがけず敵に囲まれても応戦しながら退却する。自動戦闘モードも，その辺をうろついている怪物相手ならば十分使えるものになる。

もうひとつ，プレイヤーの手間とヒマの負担軽減になってくれるのがオートマッピング機能。歩いたところをきちんと記録してくれる。PC-9801版ではシナリオ集の中で追加された機能だが，X68000では最初から装備している。これがあると，地図を完成させるつもりであちこち旅していれば，自然とゲームが進行してしまう。落とし穴のような見過ごしやすいものも，マップを見るとキチンと書かれているので，気づかなかったがためににっちもさっちもいかなくなるということもない。これがあるとないとではゲームに対する感想がだいぶ違う。オートマッピングが最初から使えるX68000ユーザーは素直に喜ぼう。



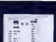
X68000用　5″2HD3枚組　8,800円(税別)
システムソフト　☎092(752)5278

マルチシナリオだ

最近のRPG，特にアクティブRPGのシナリオはやたら大作指向だ。「世界の破滅を救うのは君だ！」「今解き明かされるナントカの謎！」感動するまでは帰さんもんねという感涙の振り子攻撃である。

ブルトン・レイのシナリオはシンプルで短い。童話や昔話程度の素朴なものだ。なんせブルトン・レイというタイトル自体，中世にイギリスからフランスブルターニュ地方にかけて流行った吟遊詩人の歌物語のことだそうである。まあそんなコンセプトだからして話の規模は小さいのも納得できる。その代わりに，5本のシナリオが用意されており，現在発売されているシナリオ集も合わせると30本以上のシナリオが遊べる。

また，ひとつのシナリオを終了したキャラクターを別のシナリオに転送することもできる。しかもそれはPC-9801用のユーザーディスクからであってもかまわない（ただし，キャラの顔が化けることがある）。さらにシナリオディスクまでPC-9801版と共用できるので，X68000用ブルトン・レイを手に入れさえすれば，シナリオ集は移植を待たなくても従前のもので遊ぶことができる。移植で待たされることの多いX68000のユーザーにとっては嬉しい話だ。その代わり，自[illegible]ユーザーディスクへコピーするという，ちょっと面倒臭い手順を踏まなければならないが。



RPGとしてのルール

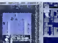
経験やステータスもしっかりしている。もちろんT&TやAD&DのようなヘビーデューティなRPGよりは簡素だが，シナリオのサイズには相応だ。

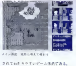
武器や防具にはS・M・L・LLの体格があって，体格の合わないものは使えない。またキャラクターの"力強さ"によって武器や防具をどのくらい持てるかが決まる。

戦闘能力は，攻撃力/命中度と防御力/回避度の要素から成っていて，そのキャラクターの戦闘能力をちゃんと発揮させようと思ったら，キャラクターのレベルにあったものを用意してやらなければならない。たとえ宝箱から「炎細の剣」なんて強そうな武器を手に入れたとしても，キャラクターに使いこなすだけの能力がなければ，かえってその重さのために防御力/回避度が落ちてしまったりする。こういったキャラクターに対するまめなケアもRPGの楽しみのうち。武器屋に行くのが楽しくなろうというもんである。敵キャラクターにもこういった戦闘能力に基づいた個性があって，戦闘時にちゃんとお互いのステータスが生かされているのがわかる。

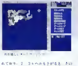
魔法は神官が使うもの。敵を眠らせたり混乱させるもの，相手の能力を下げるもの，攻撃を抑えるものの4系統，それぞれ6種類ずつあって，合計24種類がある。もっとも，魔法がからきしダメなキャラクターでも，シナリオはクリアできるからご心配なく。

これらの特徴は，やはり複数シナリオ制のために生じたものが多いようだ。特定のシナリオに依存した演出やルールは作れないし，シナリオを作りやすくするためにステータスその他のデータもわかりやすくまとめなければならない。斬新なところはないが，そのぶん自動化などのまっとうな部分を高品質に仕上げている。非常に洗練されてわかりやすいゲーム体系である。





メイン画面　地形も考えて戦おう



ゲームの評価は

X68000のRPGといえば海外の移植大型RPGか，アクティブRPGしかなかったので，どちらにも属さないブルトン・レイは非常に新鮮に映った。シナリオは短い代わりに適度なテンポで進んでいく。アクティブRPGの「壮大で一本道なシナリオ」とどっちがよいと聞かれたら，僕は迷わず前者を選ぶ。

べつにシナリオが短いだけで気に入っているわけではない。ブルトン・レイでは「西の洞に行ってみるとよい」といわれて，洞に向かって旅をしている感が楽しめるのである。自動移動モードを使いながら歩き回り，夜になったら寝る。敵が襲ってきたら戦う。この"当たり前の行動"をしている時間のバランスがとれているせいだろう。無駄な手順を踏まされている気はしない。ストーリーでわざわざプレイヤーのやる気を誘わなくても，あちこち世界を見て回って，マップ作りを進めるだけでも十分楽しめる。だから話が単純でも飽きないのだ。たとえあとで重要になる所を先に発見してしまったとしても，ちょっとのぞいて帰ってくることが立派な冒険である。

よくRPGの弊害としていわれる経験値稼ぎもない。スタート時にゲームの難易度に合わせてキャラクターのレベルが設定されており，2，3レベルも上がると，その辺にいる怪物はゲームの障害ではなくなる。一定レベル以上でないと倒せないボスキャラもいない。経験値がゲーム展開の足かせになっていないのである。

仮にブルトン・レイの弱点を挙げるとすれば，プログラム的な面である。まずX68000版はPC-9801版の画面をそのまま持ってきて極力同じにしてある。僕は高解像度モードで画面を見るのは賛成だが，PC-9801で有効だったラインスクロールが，同じようにX68000ユーザーに通用するのかに関しては疑問が残る。確かに狭くなったときの画面はPC-9801に比べてはるかに見やすいが，さらに目立つ部分でX68000ユーザーの満足感を刺激するような部分があってほしい。

また，PC-9801と比べてもパーティを組むようになると部分的に動きがカクカクしてくるし，マップを書き換えるときに時間を食ったりする。X68000ユーザーがこういうところに向ける目は，ものすごく厳しいのでちょっと心配だ。

しかし，それもこのゲームの魅力に比べれば微々たるものだ。遊べる時間やその質を考えればコストパフォーマンスはまちがいなく高い。コンピュータを始めたばかりで，RPGのイロハがよくわからない人，X68000用のゲームは過剰な演出が多くてという人には自信を持っておすすめできる。

ぜーんぶまとめて

カクカクした動きを気にしなければ素直にいいゲーム。身構えて始めなければならない大作が多いなかにあって「ちょっとやってみようかな」とつい手を伸ばす気になれるRPGは，ブルトン・レイくらいのものだ。マップをもうちょっと完成させてみようかなと歩き回っていれば，自然にキャラクターも育つしストーリーも進む。いつの間にかすっかりハマってしまっているというわけだ。ゲームはバランスだなあと思わせる一作。

シナリオ集には，ややレベルの高いものをあつめたVol.1と，ユーザーが作ったひとクセもふたクセもありそうなVol.2が用意されていて，これまたどんなドラマを見せてくれるか楽しみだ。

総合評価 (0〜10)	
操作性	★★★★★★
ビジュアル	★★★★★★★
音楽	★★★★★
シナリオ	★★★★★★★★
ゲームバランス	★★★★★★★★★
マニュアル	★★★★★★★★★
熱中度	★★★★★★★★★

●ひとこと　シナリオエディタも出してね

AFTER REVIEW

諸々の事情でしばらくお休みしていた「AFTER REVIEW」ですが，今回から復活です。復活第1弾記念ということで(?)，今回はシムシティーだけをドンと取り上げます。発売と同時に飛ぶように売れたそうで，すぐには買えなかった人もいるかもね。



シムシティー

▶シムシティーは楽しんでやっているんだけれど，人口40万人なんてとてもできない。39万人で止まっている。できるだけ陸が多いものを選んだつもりなんだけど。
岡田 政宣(19)東京都

▶シムシティーの面白いところは，メガロポリスにすることよりは1歩手前で災害の設定をして，住人の逃げるところを見ることです。やはり，みんなも同じ考えでしょう。
上原 慎平(20)神奈川県

▶ぼくの街は，いま27万人ほどである。3万人を超えるあたりで，このままでは人口が増えないと悟り，大改造(?)をしているうちに，街がきらいになってやっていない。交通渋滞の解消のために，道路を全撤去してしまう自分がいやになった。
竹田 文雄(25)静岡県

▶シムシティーはスイッチをつけて起動すると楽しい。コマンドモードで

A	SIM/H	ステージ細かいモード
	/HL	(24KHz)
	/A	いきなり新しい町作り
	/LJ	なぜかバスエラー
	/LE	

最後のは日本語と英語の切り替えだと思うが。
渡辺 篤志(17)滋賀県

▶シムシティー楽しいですね。でも不満もあります。斜めの道が作れない！ 地下鉄が作れない！ 遊園地は？ ディズニーランドは？ 都市高速道路がほしい。最後に「市役所はどこ？」でした。
坂倉 麻一(21)愛知県

▶ポピュラスはプレイしている自分が多少偉くなる。シムシティーはプレイしている自分が知的に思える。
今門 修司(19)兵庫県

▶将来はなかなか。かわいしくていい。でもいく
田中 哲(17)滋賀県

▶区画整理するよりも，住民の要求のままに拡張したほうが人口が増えるなど，妙に現実っぽいところがいい。
戸田 秀範(21)新潟県

▶街づくりが楽しい。しかし，曲がった道を見ると直したくなってしまう。
石井 久(18)千葉県


Big City……メルトダウンしているぞ

▶下手な社会科の授業を聞くよりずっとためになるし面白い。思わず自民党に味方したくなってしまう。
芳賀 寛(16)福井県

▶自分で作ってみたいと夢に見ていた街が作れるから。
岩崎 正道(18)熊本県

▶ぼーっと見ているだけで進んでいくのがいい。ついでに街を災害で壊すのも好き。アブナイ？
山本 昭敏(22)神奈川県

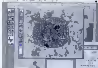
▶基本的に勝敗がないので腹もたたない。しかし，この街の市長は不老不死か？ 私なんか，かれこれ950年も生きているぞ。
岡田 伸(22)京都府

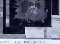
▶交通機関は鉄道だけに命をかけて，発電所はすべて原子力発電所。なんかあぶないですね。
水口 仁郎(21)岡山県

▶自分の作っている街がしだいに大きくなって，車なんかがたくさん走っていくようになっているのを見るとなんとなくいい。
相谷 健(20)石川県

▶大学で都市計画をやっているので。
和田 公生(22)福岡県

▶人間の社会生活が営まれるうえでの集団心理の重大さと，人間の環境への依存性がよくわかるから。
原田 真也(20)静岡県


Rec City……計画的に，規則的に


Happy…… 本当に幸せかな？

▶アイデアをゲームにする際のモデル化がすばらしい。
橋本 忍(19)埼玉県

▶試験が近かったのに，誘惑に負けてシムシティーを買ってしまいました。家に帰ってマニュアルとにらめっこしながら，おそるおそるやっていたところ，突然「メルトダウン」しました。初めてパッケージを開けてから20分後のことでした。誰かこの記録に勝てる人はいませんか。
村上 学(19)福岡県

▶神様はちと荷が重すぎるので。
志賀 (17)愛知県

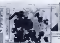
▶日本風にアレンジしたバージョンがほしい。山とか田んぼとかあって，埋め立てができて，地下街があって，また道を作るとなぜか車が集まるとか(笑)。
南又 健(21)栃木県

▶シムシティーの対戦版ってないかな。ひとりは普通にやって，もうひとりは怪獣を出したり，飛行機を落としたり……。バンゲリングベイみたい。
伊藤 龍(18)三重県

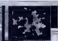
▶ゲームというメディアに対する一種の幻想が壊しられるから。
角倉 博之(21)鹿児島県

▶ここも面白い。エントレスなところがいい。
黒須 政弘(22)愛知県

▶市長の気分が味わえるから。本物よりかな


Bad C…… メルトダウン 悲惨な結果


Los…… 結構あ？からかも


SouthPark…… その名にふさわしい

……大変そう。
前田 秀樹(16)京都府

▶シムシティーしてプレイヤーは市長ってことになっている。でも市長というよりはなんちゅーか都市計画屋さんの気分だな。緑を増やして，治安をよくして，ゆとりある交通をと「うーん，こりゃいい街だ，増税も通しだな」なんて悦に入っているのに，ちょっと税率を上げると，もう住民のうるさいこと。税金が高いだの，地価が高いだの，まったくこいつらときたらものの価値を解さないんだから。
まあ，人まかりは禁物ってこと。100年以上もの時間をかけて都市とともに生きるのだ。その間，都市が成長する過程で住民の価値観は変わっていく。そう，工場が町の希望だった時代から，エコロジストがはびこる時代へと。これがシムシティーのすごいところだな。時代ごとにセーブしてあとで歴史を振り返ってみるのも一興かも。
古村 (27)東京都

シムシティーに対する読者の評価はいつもよく，さすがに名作という感じですね。また，ポピュラスと比べる人もかなりいて，やはり永遠のライバルというところでしょうか。ちなみに周りの写真はイマジニアからいただいた，作者のウィル・ライトさんが作った街です(Mac版)。

X68000用	5"2HD版	8,800円[税別]
イマジニア		☎03(343)1044

発売中のソフト

★レインフォーサー　ザインソフト
X68000用　5"2HD版6枚組　8,800円[税別]
★ダイナマイトデューク　ヘルツ
X68000用　5"2HD版3枚組　8,800円[税別]
★ザ スーパーラスベガス　日本デクスタ
X68000用　5"2HD版2枚組　12,800円[税別]
★ブルトンレイ　システムソフト
X68000用　5"2HD版3枚組　8,800円[税別]
★ノル・フィース　ウルフチーム
X68000用　5"2HD版5枚組　8,800円[税別]
★銀河英雄伝説II　ボーステック
X68000用　5"2HD版4枚組　9,800円[税別]
★楽園は遠に　アートディンク
X68000用　5"2HD版3枚組　9,800円[税別]
★イメージファイト　アイレム
X68000用　5"2HD版2枚組　5,700円[税別]
★続ダンジョンマスター カオスの逆襲
ビクター音楽産業
X68000用　5"2HD版2枚組　8,800円[税別]
★リングマスターII　ホビージャパン
X68000用　5"2HD版　8,800円[税別]
★スペースローグ　ウェーブトレイン
X68000用　5"2HD版　9,800円[税別]
★D―狼州戦士伝―　ウルフチーム
X68000用　5"2HD版3枚組　12,800円[税別]
★シュヴァルツシルト　工画堂スタジオ
X68000用　5"2HD版2枚組　12,800円[税別]
★エメラルドドラゴン　グローディア
X68000用　5"2HD版4枚組　9,800円[税別]
★ぴくせる組　ブラザー工業
X68000用　5"2HD版　4,800円[税別]

新作情報

★DRAKKHEN　EPIC/SONY RECORDS
X68000用　5"2HD版　価格未定
★ワールドスタジアム　SPS
X68000用　5"2HD版2枚組　8,800円[税別]
★マーブルマッドネス　ホームデータ
X68000用　5"2HD版　価格未定
★ファンタジーIV　スタークラフト
X68000用　5"2HD版　9,800円[税別]
★中華大仙　シャープ
X68000用　5"2HD版　価格未定
★エイリアンシンドローム　電波新聞社
X68000用　5"2HD版　価格未定
★アトミック・ロボキッド　システムサコム
X68000用　5"2HD版2枚組　8,800円[税別]
★プリンス オブ ペルシャ　ブローダーバンドジャパン
X68000用　5"2HD版　価格未定
★パロディウスだ！　コナミ
X68000用　5"2HD版　価格未定
★生中継68　コナミ
X68000用　5"2HD版　価格未定
★遥かなるオーガスタ　ティーアンドイーソフト
X68000用　5"2HD版2枚組　12,800円[税別]
★ラプラスの魔　エム・エー・シー・ハミングバード
X68000用　5"2HD版3枚組　価格未定

クリスマスにデモを

Komura Satoshi 古村 聡



Illustration T.Takahashi

クリスマス近し，ということで今月はなんと3本立てです。1本目と2本目が1組でX68000用外字ユーティリティ，もう1本がおなじみの人のデモです。あれれ，実質的にはふたつということかね。

と1．〜〜 私が恐怖の宴会男（で）であります

いやあ 飲んだ飲んだ。実はライター同士の飲み会があって イラストでおなじみの高橋君やあの"CARD FNC"の毛内さんたちと飲んできたんですよ。ああ 気持ちいいな。とっとらでも持ってきなさい。私は誰のイッキでも受けるぞ！

突然，話は変わります（しれぱっかし）。せっかく足を洗いかけていたのに，また少女まんが買っちゃったんです。わかつきめぐみさんの"So What?"というやつ本屋で1冊ぱらぱらっとめくってみたら面白そうで，気がついたら新宿の紀伊國屋で（実は初めて）残りの2巻から6巻まで買ってしまった。せっかく買わないようにしていたのに……。ということで，なんかほかにおもしろい少女まんがないですか？ ついでにわかつきめぐみさんがほかにどんなのを描いてるか教えてくれるのもうれしいです

外字の使い回し

ではーさまにはまじめにお仕事でもしましょうかね。今月の1本目と2本目は外字ユーティリティ「USK2WP.BAS」，「WP2USK.BAS」です

USK2WP.BAS & WP2USK.BAS for X68000

(X-BASIC)

東京都 木村哲也

この2本のプログラムは「USKCG.SYSとワープロのユーザーフォントとの外字データを変換する」ためのユーティリティです。8月号でもあったように USKCG.SYSには割といろいろな外字がありますが これらはワープロでは使えません。単にファイル名が違うだけではなく ファイル内でのデータの格納のしかたが違うのです。また外字はワープロで定義すると 24×24を作るだけで16×16のものもできるので こちらを使ったほうがいい場合もあります こで便利なのがこれらのツールなのです

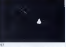
で 使い方。その名のとおり「USK2WP.BAS」（2はtoと思ってね）がUSKCGからUSERFONTを作るためのプログラム，「WP2USK.BAS」がUSEREFONTとUSKCGを作るためのプログラムです

まず カレントディレクトリにUSK2WPのときはUSKCG.SYSを WP2USKのときはUSERFONT.DATを置いてRUNしてください（ファイル名が自動的に指定されますのでカレントにUSKCGあるいはワープロのUSERFONTがないときは"ドライブ（ディレクトリ）名"でカレントを移動するようにしてください）。無事に「終了しました」との表示が出たらデータが変換されています。

おっと 注意。このプログラムはバックアップファイルを作成するといったことはしていないので取り扱いには十分注意してください。

泊たな BASICを使った実用ユーティリティ。私はうれしいぞ。そう「BASICという言語はスピードさえ気にしなければ 結構いろいろと作れるものなのであります」と思っている私としては非常にうれしいことであります。コンパイルしちゃえばスピードも気にしなくっていいということでX-BASICってのはおいしいですよね。本当に。

ただ 私はWP-Xは使えない人なので（普段からASKにメチャクチャなキー割り付けをしているのでWP-Xは手が受けつけて



COMET

それ……と残念……ますが WP-Xを使っている人にはきっと便利なプログラムに違いあるまい。うむうむ

そうそう Cコンパイラを持っている人はHuman68kの外部コマンドにしてみるときっと便利でしょうね。そうすれば，BASICを立ち上げなくても使えますし，FORMATやDIRなんかと同じ感覚で使えますからね。え？ BASICで書いたものをコンパイルすると 画面モードが512×512になるし ファンクションキーも変わるから BASICで書いたっていうのがモロにばれちゃってほかの外部コマンドと同じように使えない？

それにはですね

```
bc ファイル名.bas
```

でまずCのリストに変換する。それから，

```
#include < … >
```

と書いてあるところに

```
#include <doslib.h>
#include <stdlib.h>
```

を加えて

```
b_init()
```

を削る。そして，

```
b_exit(0);
```

を

```
C_CURON();
exit(EXIT_SUCCESS);
```

に書き換える。早い話，512×512ドットモードの画面にする部分と そこから戻る部分を取っただけなんだけど，これで一応，「Cで書いたんだもんね どうだすごいだろう」という顔ができます（おいおい）。ちょうどいいサンプルもあるし，XCを買った方はぜひとも試していただきたいな と思う（で）なのでありました。じゃんじゃん。

あ，そうだ，XCver.2.0を使ってコンパイルした場合はwarningが大量に出ますがこれは気にしないでそのまま使ってOKです。

さあ この手でBASICでガシガシと外部コマンドを作ってしまおう。そしてショートプロに送るのだ。

もうすぐクリスマスなのね

さてさて，では今月ラストの3本目。もう常連になってしまった太田さんの作品でX-BASIC用デモプログラム「COMET.BAS」です。

COMET.BAS for X68000

(X-BASIC)

東京都 太田敏郎

太田さんの投稿原稿曰く「これは自信ないなあ。でも友達が面白いと言ってくれたので送ろう。フロッピーディスクは1Mバイトもあることだし，ぱん」

簡潔なご説明ありがとうございました。実はこれ 一緒に送られてきたほかのプログラムが投稿用のプログラムでこのプログラムはおまけだったのですね。まあ，世の中とは得てしてこんなものなのでありました。うむうむ（選んでる私の性格が悪いだけという話も……）。

おっとプログラムの話ね。

画面が星空になり三角錐が現れ，そして彗星がやってきます。皆さんにはぜひ部屋の電気を消して見ていただきたい。美しい……， フフッ，デモプログラムってどうしても説明が短くなっちゃうんだよね。あんまり説明しちゃうと打ち込んだときの感動が半減しちゃうし……。これじゃ，おまんまの食いあげだよ，うるうる。

いやそれにしても なんか前々作の「夜見てはいけないデモ」のときも前作の「かべくずし」のときも言ったような気もするのですけど，これだけ短いプログラムでこんなものを作ってしまう太田さん さすがですね，しかも，今回にいたってはBASICの1画面プログラムですからね（WIDTH96だとぴったり1画面におさまる）。たいしたもんです。読みやすいプログラムのまま1画面でおさめてしまうのはまさしくアイデアとプログラムの総合勝負といえましょう（私なんかすぐ長くなっちゃうもんな，え？ ハンズのプログラムのリストが長すぎるって，はい，反省しております）。

次は1行ショートプログラムに挑戦か？

うーん 何本載せればページが埋まるんだ。それって……。でもあんまり読みにくいショートプロだったらちょっとぐらい大

リスト3 USK2WP.BAS

```
100 /* FROM USKCG TO USERFONT(WP)
110 /*                TRNS UTILITY
120 int ch,fp1,fp2
130 dim char lim(2),lln(74),nl(74)
140 str fname1,fname2,strbuf
150 fname1="USKCG.SYS"
160 fname2="USERFONT.DAT"
170 fp1=fopen(fname1,"r")
180 fp2=fopen(fname2,"c")
190 if fp1<0 or fp2<0 then print"エラーが発生しました"
200 while(fp1>=0 and fp2>=0)
210 print"USKCG.SYSをWP-Xの外字データに変換します"
220 print"何かキーを押してください"
230 strbuf=inkey$
240 print"作業中です"
250 /*     最初 34バイト 書き込む
260 fread(lln,34,fp1)
270 print ".";
280 /*     FF FF がでるまで16×16を読み書き
290 while not feof(fp1)
300  fread(lim,2,fp1)
310  ch=ch+1
320  if lim(0)<>255 and lim(1)<>255 then fread(lln,32,fp1) els
e break
330  fwrite(lim,2,fp2)
340  fwrite(lln,32,fp2)
350 print ".";
360 endwhile
370 /*     余った部分に 00 を書き込む
380 disk(&H76,ch,32)
390 disk(&H77,1,32)
400 print ".";
410 /*     FF FF を書く
420 fwrite(lim,2,fp2)
430 /*24×24(74バイト)を16×16と同じ数だけ読み書き
440 for i=1 to ch-1
450  fread(lln,74,fp1)
460  fwrite(lln,74,fp2)
470 print ".";
480 next
490 /*     余った部分に 00 を書き込む
500 disk(&H76,ch,72)
510 disk(&H77,1,72)
520 /*
530 fclose(fp1)
540 fclose(fp2)
550 fp1=-1
560 print
570 print"終了しました"
580 endwhile
590 end
600 /*
610 /*
620 func disk(x1,x2,x3)
630  lln(1)=x1
640    for i=x2 to 94
650        lim(0)=&H21+i-1
660        fwrite(lim,2,fp2)
670        fwrite(nl,x3,fp2)
680    next
690 endfunc
```



さくても読みやすいプログラムのほうを重視することが多いから気をつけてね。

三角陣に向かって飛んでくる彗星の軌道がキュベレーのファンネルみたい。行け当たれ、ピュンるんてね。それにしてもきれいですね。夜空に彗星なんて。クリスマスが近づくいまの季節にぴったりじゃありませんか　ロマンチックで。

ああ。もうすぐクリスマスか。クリスマスといえば　飲み会！　そのあとは忘年会　その次が新年会じゃない!!。飲んでばっかりでうれしいなっと。あ　なんか頭痛くなってきた　…二日酔いかな。

じゃ、そういうことでまた来月。

★酔いかなしよの部屋（困惑質問編）
Q. 8月号のショートプロぱーてぃに載っていた「なさけない顔」をGCCでコンパイルしたところ　線が1本しか出ないんです　何度も打ち間違いがないか確かめました　そして　よくわからんとあきらめたのです　2、3日して、突然、XCでコンパイルしてみたらいいのでは、と思って早速試してみたらうまくいきました　結局GCCだと違うということなんでしょうか。

A. 線を言うと私もどう説明していいのかわからないんですが　とりあえず説明してみます。ラインは本当に1本しか描いていません。というのも　このプログラムではrnd( )関数を使って円を描くように少しずつ0度から360度まで角度を増えて線を引いているんですが　そのリスト中の

rnd ( ) * 10

という部分で　0から10未満の値が返ってくるはずなのに、GCCでは

2129999528

という　よくわからないばかでっかい数値が返ってきてしまうのです　(XCのライブラリマニュアルによればrnd関数は0から1までの範囲で乱数を返す関数です)　で　1本線を引くと次が360度を越えてしまうのでもう線が引かれない　つまり線が1本しか引かれないのです。

また　XCでもver2.0になるとGCCと同様に線は1本しか引かれません　これはどうも版数の違いによるもののようです。GCCあるいはXCのver2.0をお使いの方は　インクルードするもののなかに　rnd( )が定義してある

#include <basic.h>

を付け加えるとうまくいくようです。

リスト1　WP2USK.BAS

```
100 /* FROM UHERFONT(WP) TO USKCG
110 /*                 TRANS UTILITY
120 int fp1,fp2
130 dim char lin(74),nl(74)
140 str fname1,fname2,strbuf
150 fname1="UHERFONT.DAT"
160 fname2="USKCG.SYS"
170 print"WP-Xの外字ファイルをUSKCG.SYSに変換します"
180 print"何かキーを押してください"
190 strbuf=inkey$
200 print"作業中です"
210 fp2=fopen(fname2,"c")
220 fp1=fopen(fname1,"r")
230 if (fp1<0 or fp2<0) then print"エラーが発生しました"
240 while(fp1>=0 and fp2>=0)
250 /*    最初の34バイト読みとばす
260 fwrite(nl,34,fp2)
270 /*    16×16 (34バイト) 84回読み書き
280 for i=1 to 84
290  fread(lin,34,fp1)
300 print".";
310  fwrite(lin,34,fp2)
320 next
330 /*    FF FF がでるまで読みとばす
340 while lin(0)<>255 and lin(1)<>255
350  fread(lin,2,fp1)
360 endwhile
370 /*    FF FF を書き込む
380 fwrite(lin,2,fp2)
390 /*    24×24 (74バイト) 84回読み書き
400 for i=1 to 84
410  fread(lin,74,fp1)
420  fwrite(lin,74,fp2)
430 print".";
440 next
450 /*
460 fclose(fp1)
470 fclose(fp2)
480 print
490 print"終了しました"
500 endwhile
510 end
```

リスト2　COMET.BAS

```
10 screen 0,2,1,1:randomize(val(right$(time$,2))*123)
20 int nx(8),ny(8),cd(5),c=1,no,dx,dy,vx,vy,mx,my,sx,sy
30 apage(1):for i=0 to 92:pset(rnd()*256,rnd()*256,rnd()*256)
next:apage(0)
40 while 1
50   sx=abs(mx-dx):if sx>8 then sx=8
60   sy=abs(my-dy):if sy>8 then sy=8
70   vx=vx-(vx<-sx)*(vx>sx)(-(dx>mx)*(dx<mx)
80   vy=vy-(vy<-sy)*(vy>sy)(-(dy>my)+(dy<my)
90   mx=mx+vx:my=my+vy
100  line(nx(no),ny(no),mx,my,c)
110  no=no+1:if no=7 then no=0
120  l=no+1:if l=7 then l=0
130  line(nx(no),ny(no),nx(l),ny(l),0)
140  nx(no)=mx:ny(no)=my
150  for i=0 to 5
160    palet(c-cd(i))
170    c=c-1:if c=0 then c=6
180  next
190  c=c+1:if c=7 then c=1
200  if mx=dx or my=dy then continue
210  apage(1)
220  a=rnd()*192:for i=0 to 5:cd(i)=hsv(a,31,31-i*5):next
230  for i=0 to 30:circle(dx,dy,27,0,-240-i,-300+i,400):next
240  dx=rnd()*216+20:dy=rnd()*225:a=-8*int(rnd()*31)+8
250  for i=0 to 30:circle(dx,dy,27,a+1/4,-240-i,-300+i,400):n
ext
260  apage(0)
270 endwhile
```

(で)のぱーてぃハンズ第2部――（その2）

どもども　お待たせいたしました。一部の方お待ちかねの(で)のぱーてぃハンズでございます

はい　先月号でダンジョンの壁のパターンが描けたんで今度はこいつを使って少しずつ迷路プログラムっぽくしていくんですね　先月号でいうと

「2) 自分から見える壁を読んでいし壁を描かせるルーチン」

の部分を作っていくわけです　うーん　ボトムアップ（といいつつリストはトップダウン）さかったりするけど　気にしないでね。

さてさて　それでは自分から見えるっていうのはいったいどういうことなんでしょう。さて、いま何が見えてますか？　私は　（あ　編集さんの怖い顔）目をそらしてと　ついでにバリケートを構築して　はいはいっ　いますぐ原稿あげますからっ　バリケードを越えてこないで　あははは　つまり　ものが見えるには

「自分の視界の範囲にある」

「じゃまものに遮られていない」

という2つの条件が必要になってくるわけですね（まあ　当たり前ですが）　まず自分の視界なるものを作っていきます　はい。

目の前のものは？

さてさてさて　自分の四方まで360度展開で見えるとなってても面白いんだけど　世の中そうはいかない　まして今回のプログラムはダンジョンのパターンをすべて手で描き出してしまったのでそんなことはのぞむべくもないというわけです

そんでです、先月の図2を引っ張り出してきてー　、自分からどの範囲のものが見えるんでしたっけ？　まず　自分の前と左右は見えるで　ひとつ　ふたつー　見は4つまで見えるんですよね（サイドの壁は3つだけど、こちらを向いている壁が4つ先まであるでしょ）という、とは、4×3の大きさのマップを作ってやればいいわけですね。つまり　これが自分の視界ということになるわけです。

配列なのである

さて。と　今度はマップを作るわけですね。でも　これは方眼紙ではなくてなんとかしてプログラム上にマップを表現しなくちゃいけない　マップを表現するのに都合のいいもの　文字がいくつもあって　それがたくさんつながっていて……　そう　配列なんです　要素の配列を作ってその中に1とか0とか書いて　そこが空いているとか壁になっているとかを表すようにすればいいのです

ここでは4×3のマップなので

int ary(11)=｛

｝

として　整数型の変数を12個くっつけた1次元配列を作ります。4×3なんだから2次元配列で（0　3）にすればいいじゃないかっていう声も聞こえてきそうですがそこは私の趣味です。気にしないように。

んで　配列の添字　つまりary(1)とかの1の部分とマップ上での位置との対応なんですが

0	1	2	3
4	5	6	7
8	9	10	11

（数字が配列の添字　位置がマップ上に対応）

というようにします。つまり自分はary(4)にいて自分のすぐ前がary(8)　左はary(0)で目の前がary(5)　その先がary(6)　ary(7)ということになるんです

さて　ガンガン進めていきましょう　入れものが決まったら　次は中身

配列が決まったわけですけど　整数の配列ですから32ビットの範囲何ということで　2684354551日本全国民に1人ひとり番号をふってもまだ半分　（ 種類のものが入れられますね。いろいろなイベントを…って定義できるわけです　たとえば上に通し番号とか　隠れ部屋の情報とか、

……ではとりあえず

0　何もない

1　壁

2　赤い壁（通り抜けられる）

の3つを決めておきます。では、画面に壁を描くときには、

0　何も描かない

1　白で壁を描く

2　赤で壁を描く

と壁を描くときに色分けするようにしてあげてるんです

つまり　そこにあるものの種類から壁の種類を返す関数を作る必要があるわけですね。

よし　マップに関しては決まったと　あとは描いていくアルゴリズムかな

来月につづく

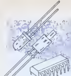
……と思ったら　もうページがないじゃないか！　今月もリストが長いしなあ　1ページを超えちゃうとなんとなくまとまりがないし……、うるうる　壁を描くアルゴリズムの解説、そしてプログラムを作るところまでいこうと思ってたのに　なんてこったい

ということで　掲載するプログラムのほうは来月分のマップ（配列）上のものを見て壁を描くところまで載せておきます（だってもったいないんだもん）で　来月はこの壁を描くアルゴリズムとプログラムの解説をしていきます。それまで配列にいろいろ入れて遊んでみてください

では　来月またこのOh!Xで。つづくっ！（と編集を立てて去る）

リスト

[illegible]





ハードウェア工作入門〈7〉

センサー回路その1

Kazuhiko Misawa
三沢 和彦

今回からA/Dコンバータを利用した身近でより実用的な応用例に挑戦することにします。取り上げるテーマはセンサー回路です。まず1回目は理論編ということでセンサーの種類や具体的な働きを整理することから始めてみましょう。

前回までにA/Dコンバータボードを完成させ、ごく簡単な応用としてアナログジョイスティックを試してみました。しかしそれだけでは A/Dコンバータの本領を発揮させていません。今月からは、A/Dコンバータの本格的な応用として、いろいろな種類のセンサーをこのA/Dコンバータボードに接続してみることにします。

センサーとは何か

10月号のA/Dコンバータの解説の中で、機械が人間の感覚の代わりに自然界の量を計測するシステムについて述べました。そして、このようなシステムにおいては、光の明暗、音の強弱、温度の高低などのアナログ量は、すべて電圧に変換されたのちにA/Dコンバータに入力されます。この、自然界のアナログ量を電圧に変換する装置をセンサーといいます。

センサーはもともと人間（や他の生物）の感覚器官を代替し、その機能を拡張したものを意図して作られましたから、基本的なセンサーとして、人間の五感（視覚、聴覚、触覚、味覚、嗅覚）に対応したものがあります。しかし、人間の感覚には、自然界の物理量を「感知する」ことに加え、感知した信号から外界の対象についての情報を「認識する」機能があります。

たとえば、視覚に関していえば、目に入ってきた光が網膜上に結像を結ぶとき、網膜上の各点における細胞が光の色と強度とを検出する段階が「感知」にあたります。その次の段階として、網膜上の各点で感知した信号を組み合わせ、それを像として捉えることによって、対象の色や形、または位置関係を把握する段階が「認識」ということになります。

人間の感覚器官を機械的または電子回路的にシミュレートできるのはせいぜい「感知」の段階までで、「認識」の段階まで行うのには限界があります。現在「感知」の段階であれば、実用上で人間より優れた性能を示すセンサーは多くあります。視覚では光センサー、聴覚では音波センサー、触覚では圧力センサー、味覚と嗅覚ではガスセンサーなどが対応します。要するに、人間の持つ感覚機能のなかで、センサーは特定の物理量だけについて精密に計測する機能にのみ対応しているというわけです。ですから、実際の感覚機能をシミュレートするには、複数のセンサーによってより多くの物理量を計測し、それらの情報をコンピュータで処理しなければなりません。このとき、コンピュータは人工知能のシミュレートをしていることになります。

現在のセンサー技術においては、特定の物理量を精密に計測する性能は特に優れています。人間の感知できる光や音波であっても、感知可能な領域は限られていますが、センサーであれば、人間の感知できない紫外・赤外光や超音波でも高感度に検出できます。また、人間がまったく感知できない磁気や放射線などのセンサーもあります。

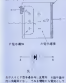


図1 フォトダイオードの構造

光が入るとP型半導体内に正電荷、N型半導体内に負電荷が生じ、それを電極から電流として取り出す

そこで次に、これらのさまざまなセンサーを整理し、それぞれのセンサーが感知できる物理量とその原理や性能について、簡単に説明していきたいと思います。

センサーのいろいろ

センサーを分類するときは、検出する物理量にしたがって分類するのが最もわかりやすいと思います。ここで説明するセンサーは光センサー（放射線センサー）、音波センサー（圧力センサー）、温度センサー、ガスセンサー、磁気センサー、位置センサーの6つです。

1） 光センサー

数あるセンサーのなかで、最もポピュラーで手軽に使えるのが光センサーでしょう。電化製品ではもう常識となっているリモコンはこの光センサーの応用そのものです。

現在使われている光センサーの大部分は、半導体を用いたフォトダイオードあるいはフォトトランジスタです。このうち、より構造の簡単なものがフォトダイオードで、これはP型半導体とN型半導体とを接合した構造をしていて、それぞれに端子が出ています（図1）。

フォトダイオードに光が当たったときにはダイオード内部のP型の部分には正の電荷が、N型の部分には負の電荷が生じ、それぞれの電荷が端子から電流として取り出せる仕組みになっています。そして、その電流の大きさ（内部に発生した電荷の量に相当する）が当たった光の強さにほぼ比例するので、光の強度を測定することができるのです。

フォトダイオードの分光感度はその材料によって決まりますが、半導体で最もよく使われているシリコンを材料とすると、波長で200～1100nmに感度があります。人間の可視光は400～700nmですから、ちょっと可視光全域をカバーすることができます。もちろん、人間の見えない近紫外光、近赤外光にも感度があることになります。また、その感度の絶対値ですが、10^{-4}～10^{5}ルクスの極めて広い範囲をカバーします。これはおおよそ夜の星明かりから真夏の日差しまでに対応しています。さらに人間の目では感じることができないくらい微弱な光を検出するには、光電子増倍管という別の仕組みのセンサーを使いますが、詳しいことはここでは省略します。

桑野氏のガイガーカウンタの製作記事で詳しく解説されていた放射線も光とともに電磁波の一種なので、放射線センサーといっても今述べた光センサーと同じものです。しかし、放射線は可視光に比べて物質の透過力が強いので、放射線の吸収力の大きい物質を選びます。このなかでシリコンよりもゲルマニウムやテルル化カドミウムといった材料のほうがよいようです。

2） 音波センサー

音波は空気を伝わってくる振動です。もし音波が物体に当たるとその振動エネルギーが物体に移って、物体も振動します。一番顕著な例が、ギター弦のチューニングをするときに、音叉をそばにおいて弦を弾くとその音が音叉に乗り移って音叉が振動し始める現象です。そこで、音波センサーはセンサー自身の力学的振動を電気信号に変える仕組みになっています。それには、圧電効果といって、歪みを受けると電圧を発生させる材料が用いられます（図2）。圧電素子にかかる歪みは音波の強さに比例し、したがって発生する電圧に比例しますので音波センサーとして使えるわけです。

よく利用される音波センサーはいわゆるマイクです。このマイクの中にも圧電効果を示す素子が入っています。その構造は、セラミックまたは半導体の圧電素子を2枚張り合わせた板の両面から端子を取り出した形になっており、板全体が振動する際に生じる板両面の電位差を検出しています。

音波センサーはマイクのように人間の可聴領域にある音声を検出するだけでなく、オートフォーカスカメラの距離計や物体の速度・加速度計あるいは潜水艦用の水中探知器などにも使われています。これは、センサー自身から音波を発信し、物体に当たってはねかえり、再びセンサーに戻ってくるまでの時間を計測することにより、そのときの音波の速度から計算して物体までの距離を測定するのが基本になっています。このときは、可聴音だと耳障りなので、40kHz程度の超音波を使っていますが、センサーそのものの基本的な仕組みは変わりません。

3） 温度センサー

温度センサーも家庭の電化製品ではよく使われています。エアコンの室温調節には欠かせません。温度センサーの大部分も半導体を用いたものです。半導体に電圧をかけたときにその中を流れる電流の大きさは温度によって変わります。一般には、温度が高いほうが電流が流れにくく、温度の逆数に対して電流値は指数関数で変化します。

実用的には、トランジスタの電流－電圧特性を利用します。トランジスタにはエミッタ、コレクタ、ベースの3つの端子があり、エミッターコレクタ間に一定電圧をかけておくと、ベースからエミッタに流す電流値に比例してコレクタからエミッタへの電流値が変わる仕組みになっています（図3）。ところが、エミッターコレクタ間電圧とベースからエミッタへの電流の両方を一定値にしたとしても、周囲の温度によってコレクタからエミッタへの電流が変化するのです。そこで、このコレクターエミッタ間の電流を測定することによって、温度が測定できるのです。最近では、さらにこの電流を電圧にかえて、しかも出力電圧値が直接摂氏温度に対応するようなICセンサーが普及しています。これによって、IC1個で特に外付け部品もなく、0～150℃の範囲が一発測定できるようになっています。

4） ガスセンサー

ガスセンサーは火災報知器などに使われていますし、工場などの安全管理にも欠かせません。また、あまり身近ではありたくないのですが、警察が飲酒運転の取り締まりを行うときのアルコールセンサーもガスセンサーの一種です。

ガスセンサーはセラミック、半導体ともに素子の表面にガスの分子が吸着したとき



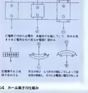


図2 圧電素子の構造

圧電素子内部の正電荷、負電荷が分離していて、形状を歪ませると電荷分布の変化が電圧で現れる

圧電素子を2枚張り合わせる

片方が伸び、もう片方が縮むことによって板全体が振動し、そのとき電極に電圧が生じる

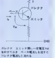


図3 トランジスタの温度特性

コレクタ－エミッタ間に一定電圧V_{CE}をかけておき、ベース電流I_Bを流すとコレクタ電流I_Cが流れる
このとき、ベース－エミッタ間電圧をV_{BE}とすると
$I_C = I_S \exp\left[\frac{qV_{BE}}{kT}\right]$ （I_S：定数）
と温度Tの関数となる

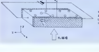


図4 ホール素子の仕組み

半導体のZ方向に電流I_1を流しておくと、Y方向の磁場Bに対して$I_1 \times B$に比例する電圧V_HがX方向に生じる

図5 アルコールセンサーの回路図

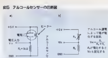


図6 光センサーの回路

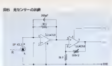
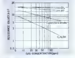


表7 Sensitivity characteristics



に電流に対する抵抗値が変わる性質を利用しています。抵抗値はセンサー周りのガスの濃度の関数になっているので、センサーに電圧をかけておき、その電流値を測定することによってガスの濃度を検出します。ただし、それぞれのセンサーは検出するガスの種類までは判別できず、水蒸気、メタンガス、アルコールガスなど大抵のガスに反応してしまいます。ガスの種類まで判別するには、ガスに対する感度の違ういくつかのセンサーを組み合わせる必要があります。

実際にガスセンサーを使う場合には、一度センサー表面に吸着したガス分子によって感度が落ちてしまうので、ヒーターで熱しながら、吸着した分子を蒸発させるように工夫しています。

5) 磁気センサー

磁力線の強さを計るセンサーには、ホール効果というものを利用した半導体ホールセンサーが一般的です。ホール効果とは、図4のように直方体状の半導体に電流を流しておき、その電流の向きと垂直に磁場をかけると、電流と磁場のどちらにも垂直な面に電圧が生じる現象のことです。しかも発生する電圧は、かけた磁場の強さに比例するので、あらかじめホール素子に一定の電流を流しておき、電圧出力端子の値を計測することによって、磁場の強さを測定することができるのです。

6) 位置センサー

位置センサーの最も簡単な例はボリュームです。前回のボリュームを使った簡易アナログジョイスティックは、この位置センサーのひとつの応用例といえます。これは、抵抗値が抵抗線の長さに比例することを利用したもので、詳しい仕組みは、前回説明したとおりです。

ところが、ボリュームを使った位置センサーには重大な欠点があります。それは位置を検出するための接点が機械的に抵抗線と接触しているということです。機械的な接点は長時間使用していると必ず磨耗します。そこで、光センサーや磁気センサーを利用した位置センサーが工夫されています。これは、抵抗線の代わりに光センサーや磁気センサーをずらりと並べておき、移動する接点に発光体や磁性体を付けておくと、接点の位置にあるセンサーだけ反応するという仕組みです。この無接触型の位置センサーが、ボリュームを利用した接触型のセンサーよりも実用的です。

センサーの実用例

では、今回の動作実習回路を図5、図6に示します。1で説明したセンサーのうち今回は光センサーとガスセンサーとを使ってみました。位置センサーは前回使いましたし、音波センサー、温度センサーおよび磁気センサーはいずれ取り上げてみようと思っています。

まずはガスセンサーを使ったアルコール検出器です。これは市販のアルコールセンサー1個で外付け部品なしの極めつけ回路です。アルコールセンサー自体の使い方は普通の抵抗器と同じですから、図5(a)の回路は図5(b)と等価です。したがって、A/Dコンバータの入力端子にかかる電圧は電源の+5Vをセンサーの抵抗Rsと負荷抵抗RLとで内分された値になります。空気中のアルコールの濃度によってセンサーの抵抗値が変わるので、それに応じて出力電圧が変わる仕組みになっています。参考までに、図7にアルコール濃度とセンサー抵抗との関係を示したグラフを挙げておきます。

次はフォトダイオードを使った光センサーです。図6を見てのとおり、LM358というIC1個を付け足しています。このLM358はオペアンプというアナログ信号の増幅器2個からなっています。このオペアンプを2段組み合わせた回路は、フォトダイオードが光を受けて出力した電流を電圧に変え、しかもそのままの電圧ではA/Dコンバータで測定するには小さすぎるので、測定可能な電圧値にまで増幅するものです。1段目は電流増幅器、2段目は非反転の可変増幅器というものになっています。

お馴染みのTTL-ICは基本的なデジタル回路をパッケージに納め、世界的な統一規格にしたがって通し番号が付けられていますが、このオペアンプは基本的なアナログ回路をパッケージ化したものです。使い方もTTL-ICと同様に、規格にしたがって端子間をつなぐだけの簡単な作業ですみます。とはいえ、アナログ回路はデジタル回路よりもさらに皆さんの馴染みのうすいものではないでしょうか。

というわけで、来月はたっぷりと1回分を割いてオペアンプによるアナログ回路入門を解説したいと思います。そのときに1で述べた「電流増幅器」、「非反転増幅器」などの言葉の意味にも触れるつもりです。さらに、フォトダイオードの使い方も一緒に説明したいと思いますので、光センサー回路の詳細は来月までお待ちください。

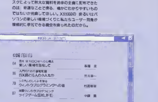
●

以上、今回はセンサー理論編として、さまざまなセンサーの種類とその仕組みについてひと通り解説してみました。もちろん文章で説明するだけでは理解しにくいと思いますが、来月はもう1回理論編で我慢してもらうことにして、再来月には製作実習に移りたいと思いますので、お楽しみに。

SX-WINDOW



SX-WINDOWとはなにか。それは誰のためにあるのか。ひとつの答えは「アプリケーションの実行環境であり、ユーザーにとってわかりやすい操作環境を提供するシステムである」ということだ。だから本来、エンドユーザーはウィンドウ上でプログラムが走るというのがどういうことかなど考える必要はない。しかし、あえて本誌の読者にはパソコンの未来のために考えてほしい。現時点でSX-WINDOW上のアプリケーションはまだ市販ソフトのレベルでは存在しない。実際のところ、ソフトメーカーに対してもまだ資料やツール類のサポートが始まったばかりの段階だ。だからそのような時期に、付録ディスクによって膨大な資料を読者の全員に配布できたのは、幸運なことである。確かにわかりやすいものではないか挑戦してほしい。X68000の、あるいはパソコンの新しい環境づくりに私たちユーザー自身が積極的に参加できる機会を得られたのだから。



X68000

SX-WINDOWへの心構え

美しい環境を目指して

Ogikubo Kei 荻窪　圭

X68000の新しい操作環境として登場したSX WINDOWですが，ようやくアプリケーションを開発できる環境が整いつつあります。そこで，ウィンドウ環境のあるべき姿を考えつつ，私たちも心構えを新たにすることにしましょう

きしむマウス
うなるハードディスク
たゆたうメモリ
SX-WINDOWがやってきた
ヤア！ ヤア！ ヤア！

てなわけで，机に肱をつき，掌に顎を乗せ，少し斜めにCRTを眺めながら，右手でマウスをなでる。おおあくびひとつと，適当なクリック。ずりずりとアイコンを動かし，慣れた手つきでアプリケーションを動かす。それが"SX-WINDOW!"と思ってはいないか。

確かに間違いはないかもしれない。ただし，それはエンドユーザーだけに許される姿だ。エンドユーザーが楽をするにはプログラマが苦労をし，プログラマが楽をするにはシステムの開発者がエンドユーザーが苦労する。ところが，エンドユーザーは楽をしているし，システムは出来上がっている。

そう，掌に顎を乗せてあくびをしている場合ではないのだ。

さもありなん，エンドユーザーに徹するのもよいが，SX WINDOWの資料も（洗練途中ではあるが）出揃ったいま，SX WINDOWワールド形成に向かっての心構えについて考えてみたい。ちゃんと白八つの煩悩を洗い，一年の計を，じゃなかった針を足元に立てるのだ。

VS. Xはなんだったのか

ビジュアルシェルが出た。アイコンをダブルクリックすると，プログラムが実行された。やったあ。

さあ，使うぞ。あれ？　マウスが使えない。どうしてだ。せっかくのマウスじゃないか。よし，マウス対応のアプリケーションを立ち上げるぞ。あれ？　スクロールバーの形が違う。ファイルがアイコンで表示されない，右ボタンが使えない，クリップボードや辞書や，ソフトウェアキーボードはどこへ行った？

どこやらのアプリケーションを責めるつもりはない。よくよく考えてみれば当たり前だったのだ。なぜなら，ビジュアルシェルもひとつのアプリケーションだったから。

しかし，われわれは期待していた。同じPRO-68Kシリーズならきっと操作性も揃って，データのやりとりも簡単にできるよね。……で，できない。

斜『するとビジュアルシェルに戻る。じゃあ，ビジュアルシェルってなにをやってるんだ。あのウィンドウはファイルアイコンを表示する以外に能はなかったのか。ファイルの中身ひとつ見る機能がないぞ。

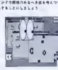
あーあ，こんなことない，コマンドシェルにしようっと。

そういう流れを多くの人がたどったに違いない。少なくとも私はそうだった。ビジュアルシェルからなにかを実行することはできても，ビジュアルシェルの上で実行することはできなかったのだ。

SX WINDOW，プログラムを起動しても消えてしまわず，どのアプリケーションも同じような画面，操作性を貫くことができ，いつだって電卓が使える。SX WINDOWの上でプログラムを実行することができるようになったのだ（というよりSX WINDOWの上で動くプログラムを書けるようになったのだ）。

せっかくだ，せっかくだから，ビジュアルシェルの轍を踏みたくない。まだろくなアプリケーションもないSX WINDOWを秩序のない乱れた世界にしたくはないだろう。

だから，次のような話が始まる。



しまかりといわれてもしかたがない
食わず嫌いは［illegible］ないのだ

1) とりあえず，とっついてみるべし

とりつかねば何も始まらない。今までの環境に慣れ［illegible］新しいSX WINDOWはとっつきにくいに違いない。しかし，慣れたものにいつまでもしがみついていたら，人間は変化を止めてしまう。変わることを恐れるようになったら，もう残りの人生は少ないと思われてもしかたがない。食わず嫌いはいけないのだ。

2) 最初は面倒だと観念すべし

ウィンドウシステムでプログラムを書くというのはおしなべて最初は面倒である。新しい規則，新しい概念，新しい用語。今まではいきなり自分のやりたいことだけを記述すればよかったのに，初期化は面倒臭いし，イベントがどうしたこうした，タスクマンの相手から，ウィンドウマンやコントロールマンうんぬん，目的のアルゴリズム以外に書くことが多すぎる気がするだろう。しかし，それらはみな慣れてしまえばワンパターンの儀礼のようなもの。さまざまな恩恵をこうむることを思えば，儀礼なんかよりずっと花も実もある。面倒な手続きは手続きのための手続きではなく，自由と秩序のための手続きなのだ。

3) 曲芸で客を沸かすのは10年早い

よくいるんだ，［illegible］マンをいちいち呼び出すより，いっそアセンブラで書いてしまったほうが早いと思うやつが。それはそれで楽しいだろうが，まだちょっと早すぎる。環境が整わないうちに環境を混乱させるアクロバットはただ邪魔なだけだ。まずは同じルールのもと，秩序を作るところから始める必要がある。

4) デザイナーたれ

SX WINDOW上で稼働するすべてのプログラムは美学を必要とする。男のロマンといってもいい。せっかくの階調モノクロで，影文字やカッコいいアイコンがあるのだ。そこにダサいやつらが紛れ込むのは楽しくない。たとえば，文字は12ドット影文字だ，装飾は淡く，ほこりまみれの単純アメリカンよりは純の屈折した美学のヨーロピアン，ちょっと斜めからカッコつけてみる姿勢がよく似合う。ひたすらカッコよく，なおかつ，自分の世界に溺れないようなセンスが重要だ。

5) 私だけの操作環境を作り出すな

せっかくのウィンドウシステムである。操作環境は標準アプリケーションの作ったものを（とりあえず）踏襲すべし。あまりにも斬新なスクロールバーなど疑問符を背中に背負ったようなものは別としてだ。左クリックでポップアップメニューが出現したり，ウィンドウクローズボタンが右下にあったり，トリプルクリックなどを駆使したりはしないように。さらに，プルダウンメニューしかないとき，ポップアップのカット＆ペーストが使えないのもいやだ。

また，せっかく，ファイル（オブジェクト）をアプリケーションウィンドウにドラッグして引き渡すというオペレーションがあるのだから，そういうところは採用する。間違っても自分のプログラムでファイル一覧をし，そこからファイルを選択させたりしないように。

クリップボードも，使えるかどうかは別にして，せっかくあるものだから，クリップボードを介したテキストのカット＆ペーストは文字入力可能なすべての局面で採用してほしい。

くれぐれも，アプリケーションひとつで「私だけの世界を作らない」ことが重要だ。アプリケーションがみんなで我を張り始めると，ウィンドウシステムの意味がひとつもなくなってしまう。

6) 恥ずかしい真似はするな!

恥ずかしい真似というのは，たとえば「さ○りん」なんて恥ずかしい名前をウィンドウに入れるとか，そういったことだ。これは恥ずかしくてダサいことなのだ。

SX WINDOWは仲間内で閉じた世界ではなく，どのアプリケーションにも均等に開かれた世界である。趣味のプログラムならともかく，少しでもSX WINDOWという環境を作り出す一端を担うのなら，恥ずかしい真似はすべきではない。子供っぽい自己顕示は笑われるだけだ。みこしを担ぐ必要もない。

7) 閉じた世界でワイワイはやめたい

SX WINDOWはとても狭い。狭いのはCRTなのだが，その狭い空間で何をするか。ただ絵を動かし遊ぶより，そこを自分の環境として，生活や作業の一端を担わせたい。

今しか笑えないふざけた娯楽より，自分を導く明日の娯楽だ。スケジューラでもタイムテーブルでもメモ帳でもデータベースでもドローイングでもいいが，SXシェルというのは作品発表の場であると同時に，エンドユーザーがなんらかの作業をする場でもあるのだ。足元を固めないと，ただ楽しいだけで終わってしまう。すべてを終わらせるのは硬直した心だ。

8) カッコつけるだけでは意味はない

で，デザインを気にして，操作性を統一して，悪さしないで，とそこまでいうからには，それなりのメリットがあるわけである。

たとえば，複数のアプリケーションを開いて，互いにデータをやりとりする。左に通信ソフトを開いて，右にワープロ。通信のログから必要なところをワープロにリアルタイムでカット＆ペースト。また，ワープロで作った文書を通信ソフトにペーストとすると，ほら簡単にアップロード。

住所録，電話番号部分を通信ソフトの電話ボタンに渡してやると，オートダイアル。いままでの環境では同じソフトハウスが全部整合性を考えて作らないとできなかったものが，SX WINDOWなら並のプログラムでも（ちゃんとつくれば）こういったデータのいったりきたりが簡単にできるのだ。

まだあるぞ。グラフィックを張り付けられる15パズルと，グラフィックエディタがあるとしよう。片や絵を作成しながら，片や絵をファイルアイコンをパズルのウィンドウにドラッグする。すると，1～15の数字の代わりにその絵が埋め込まれるとか。

一度に同じアプリケーションをいくつも起動できるのもおいしいかもしれない。ワープロを2つでもいいし，スプレッドシートを2つでも3つでもいい。

私が欲しいのは，文書整形印刷ツールだな。テキストファイルのアイコンをドラッグしてやって，ちょいちょいと設定してやると印刷してくれる。こうすると，書く道具と印刷する道具を分けることができる。

それに，ある作業をしている最中に，ちょっと別のことをしなくなったら，いままでは一度クローズするかチャイルドプロセスを起動するかしかなかったのだが，SX WINDOWならそいつのアイコンをダブルクリックしてウィンドウをまた開けばすむのだ。

ワープロで仕事をしているとき，ああ疲れたな，と立ち上げてはあるのだが途中でポーズにしてあるゲームのウィンドウをアクティブにして，ゲームやって，ゲームも疲れたな，とまたワープロに戻って仕事ってのも簡単だ。勉強するふりをして遊ぶのには欠かせない。いままでは，疲れたなといってもOPT.1＋□でテレビを見るくらいしかできなかったからね。

ほら，こう考えると，画面をいっぱいに使って作業しているとき，ダサいやつが混じっていると使えないでしょ。こういうことができるというのに，ワープロが私だけの世界を作って，変なファイル操作をしたり，通信ソフトだけが左ボタンでポップアップメニューだったりしたら困るでしょ。

そういうことなのだ。

＊

SX WINDOWがウィンドウシステムとしてどれだけ使い物になるか。果たしてMacやMS WINDOWSのように使えるアプリケーションは現れるのか。まだ未知数の部分は大きい。われわれユーザーがやることは，手探りながらでもSX WINDOWの可能性を追求していくことである。どんな可能性でも見つかれば楽しいし，そこから何かが生まれれば面白い。SX-WINDOWというのは気が向いたときに使うツールではなく，常時触る環境なのだ。その気になればなんだってできる。なんだって。



入門のための基礎知識

SX姫と15人の小人たち

Yoshida Kouichi 吉田 幸一

SX-WINDOWを支えるもの。いや、その実体は15個に分けられたマネージャたちです。マルチウィンドウ、マルチタスクに必要な処理をそれぞれ分担しています。まず、ウィンドウの基礎知識とこれらのマネージャの機能を見てみましょう。

これから先は、SX-WINDOWが発売された時からの世紀となる。その時、橋本はイ…久米は3位になるわ。…を辞任するわ。即位の礼は行われるわ、といろいろ状況は変わった。

SX-WINDOWを買ってしまった皆さん、どうしているだろうか。私が2回ほど…、すっかり音沙汰がなくて不安だったのではないだろうか。不安だったのは私も同じである。

もしかしたら、SX-WINDOWを高級なグラフィックシェル(VS X)だと思っている人もいるかもしれない。SX-WINDOWを…ような環境が出現すると…している人もいるかもしれない。SX-WINDOWなんてクソだぜ、って思っている人もいるかもしれない。まあ、なんだっていいけど。MS-WINDOWSだってVer2.11まではクソだったのだがVer3.00になって少しはまともになったように、いまのSX-WINDOWがクソだと思っても、明日のSX-WINDOWは脳天気かもしれないのでそう簡単に投げないように。投げるのはいいけどな、拾ってこられるから。捨ててはいけないぞ。ゴミ収集車に持って行かれたら復活しないから。

というわけで、SX-WINDOWの歴史は今日から始まる。昨日までは紀元前だったのだ。

SX-WINDOW登場

SX-WINDOWのようなウィンドウシステムを語る際、見る者の視点をなおざりにしてはならない。

エンドユーザーの視点と開発者の視点だ。いままで、ウィンドウシステムといえばエンドユーザーからの視点ばかりが強調されてきた。開発者にとってみれば「マウスをどう動かすと、このウィンドウが動くのか」ってことより「どのように記述すればウィンドウを動かすプログラムを書けるのか」のほうが重要なのだ。

今日から歴史は始まる、としたのは、やっと開発者の視点でもSX-WINDOWを見られるようになったからだ。つまり、SX-WINDOWの中身や規則が公開されたのである。それが付録に圧縮されて詰まっている膨大な技術資料だ。私が書くのは、そのさわり。エンドユーザーにとって重要な、目に見えない部分でのSX-WINDOWがなんとなくわかるような話。

さて、OSというものがある。パソコンの世界ではデータや文字コードの共通化やコマンドのサポートくらいしかありがたみがないように思われているが、本来は違う。大きなコンピュータの世界でのOSはもっと権力を持っており、OSを通さずにはとんな些細なプログラムでさえ動くことができないのだ。ユーザーのできるすべてのことがOSを通してなされる。そうしないとコンピュータがちゃんと動かない。OSの目的というのはそういうコンピュータ資源の管理であり、同じOSなら同じプログラムが動くというのはその次の問題だ。

パソコンがそうでないのは性格上の問題で、ちゃんとしたOSを載せるほど管理する必要のある資源もなかったし、そういうOSを載せられるほどメモリもなかった。それに、一度にひとりしか使わず、一度にひとつのプログラムしか実行されない環境だから、個々のプログラムがなにしようと、立つ鳥あとを濁さなければ構わなかったのだ。

それが、ウィンドウシステムとなるとちゃんとOSしなければならなくなる。一度にいくつかのプログラムが立ち上がったりし、ハードウェアをいろんなプログラムで共有する。そうすると、暴走するプログラムがあると、ほかのプログラムに影響が出てくる。いままではプログラムひとつにつき画面全部を好きにできたのが、ウィンドウシステムになると、自分のウィンドウしか好きにできない。こういったさまざまな制約が必要となってくる。

とりあえず、制約が増えて自由に書けなくなった代わりに、データさえ入力すればいいあとやってくれる小人がたくさんいると思えばいい。ただ、小人を動かすための手続きや言葉、動かし方を覚えるのが面倒なだけで、わかってしまえばそれほどでもない、という話だ。

このようにウィンドウシステムが複雑な処理をしてい…、プログラマがそれに対応するプログラムを書くと、皆さんが共存共栄するわけである。万歳（私は海部か…って）。Macを嫌いだという人はよく規制でがんじがらめだからという。それもひとつの考えだがね。

ウィンドウシステムとは

ふつうウィンドウシステムというと、使うことがメインであり、自分でプログラミングして使おうなどとは考えないものだ。UNIX上のX WINDOWならともかく、たとえば、「MS WINDOWS用アプリケーションをプログラミングしよう」というのは、やむにやまれぬシステムエンジニアくらいのもので、自発的にそうしようとするのはそうとう奇特な人と思われている。

にもかかわらず、SX-WINDOWの開発資料を心待ちにしていた人は多いはずだ。しばらく前のシャープの広告を見ると「…だわり続けてきたある執着がまさに帰結しようとしています」とある。開発者もこだわっていたそうだが、ユーザーはもっとこだわっていたのだ。

さて、SX-WINDOWでの開発言語はC言語またはアセンブラとなる。

特に今回はアセンブラでのアプリケーション作成を主眼とした特集だ。「どうやったらSX-WINDOWで動作するプログラムを作成できるか？」について解説するのだから、アセンブラでのプログラム作成能力は基本的な前提だ。

昨年（1990年）6月号のアンケート発表ではアセンブラプログラマが23%とCプログラマの21%をわずかに凌駕する。8ビット機ならともかく、16ビット機以上でこんなにアセンブラ野郎がいるマシンは空前絶後といっていい。

そんなアセンブラがバリバリに使える人には必要ないかもしれないが、一応ウィンドウシステムでの概念とディスク内のドキュメントを読むための用語の整理を行っておくことにする。

●ウィンドウ

画面のなかにある別の画面。ひとつの画面内にたくさんの画面を開いて、そこで別々のことをするのがマルチウィンドウだ。WP XやmicroEMACSの画面分割はタイリングマルチウィンドウ、VS XやSX-WINDOWはオーバーラップマルチウィンドウ。流行りのオーバーラップではビットマップディスプレイ、ポインティングデバイスが揃いで出てくることが多い。ビットマップとはちまたでいうグラフィック画面（ただし色は関係なし）のこと、ポインティングデバイスは画面上の任意の点を指示するための装置のうち、カーソルキーよりは使い勝手のよいものを指す。たいていはマウスのことだ。

●アクティブウィンドウ

ウィンドウがいっぱいあっても、マルチタスクしていても、ユーザーの操作を直接受け付けるのは基本的にひとつのウィンドウだけだ。これがアクティブウィンドウ。概念的にはいちばん上に重なったウィンドウ。SXシェルではなんとなく神が違うやつなので見分けることができる。

シングルタスクのMacintoshなら「アクティブ」はわかりやすいが、SX-WINDOWはマルチタスクだからタスクがアクティブでもアクティブウィンドウとは限らない。晩子さんはウィンドウの下でも働くでしょ。

●アクティベート処理

アクティブウィンドウにすること。通常、そのウィンドウがあらゆるウィンドウの上にくるように画面が書き直される。

■マルチタスク

SX-WINDOWの晩子Xが例に挙げられるが、実際には画面に出ているすべてのウィンドウでマルチタスクできる。動かないように見えるのはそのようにプログラムしてあるだけの話。

リレーの第1走者が審判員からバトンをもらう。所定の区間を走って第2走者にバトンを渡す。ただしアンカーはいない。みんな走り終わるとまた第1走者から回り始める。これが十分に高速なら、みんながバトンを持っているように見えるはず。SX-WINDOWではなにかイベントがあれば一斉に回り始める。たまに回んない競も出てくるが、なにも指定しなければマルチタスクだ。

一定時間でバトンを渡すように縛ったのがよくあるタイムシェアリングシステムというやつで、SX-WINDOWとは関係ない。時計もなしに一定時間で走れというのは無理だし。スエーデンリレーかもしれない。SX-WINDOWでは、誰でも好きなだけ走っていい。が、ひとりがあまり走りすぎるとマルチタスクに見えないので、しかるべく配慮すること。

●ヒープ

SXシステムではフリーエリアをヒープ領域とスタック領域に分けて管理する。プログラムやデータが入るのはヒープ領域だ。ヒープってのは聞き慣れない人にはまったく聞き慣れない言葉だが、元の意味は塊とかどっさりとかそんなものだ。気にしなくていい。情報処理業界ではちゃんとそれなりの意味はあるのだが、気にすることもないだろう。ダイナミックにメモリを割り付けるための領域を指す言葉だ。

当たり前だが、どのプログラムがメモリのどれだけを確保しているか、ちゃんと管理しなければならない。その単位をブロックという。ヒープ領域をブロックごとに管理しているわけだね。ブロックにはリロケータブル・ブロックとアンリロケータブル・ブロックがあって、リロケータブルというのはメモリ空間のどこに置いてもいいよっていう意味。こちらが基本。僕はここしかいやだ！という駄々をこねるやつがたくさんいたら困るのだ。このメモリーマネージャさんのおかげで、プログラマの考えることが少し減る。任せてしまえばいいから。

■クリップボード

基本的にウィンドウシステムでは各ウィンドウ間でデータをやり取りするための方法が用意されている。そのうちのひとつがMacintoshで使われているクリップボードだ。SXシェルにも用意されているが、Macintoshではテキストデータだけでなくビットマップデータも保存できるので便利。

メモリ上のデータフォーマットが統一されているならデータのポインタをこっそり教えてあげたほうが早いかもしれない。

●リソース

リソースというのは資源という意味である。が、この業界で資源というとあまりに

ウィンドウの構造

SX-WINDOWでの標準的なウィンドウの各部を見てみよう。まず、いちばん上のタイトルなどが書いてあるところは、ずばり「タイトルバー」という。タイトルバーの上で右端にある×印は「クローズボックス」。その名のとおり、ウィンドウを閉じるためのものだ。

ウィンドウの右側と下側についている部分は「新感覚のスクロールバー」でウィンドウをスクロールさせるためのもの。どう動くのかはよくわからない。

右下隅の矢印つきの四角形は「グローボックス」。ウィンドウの大きさを変えるときに使う。

ウィンドウの大きさが変わらないとわかっているときはタイトルバーの部分と本体だけのウィンドウもある。

さらにタイトルバーもなく、四角形がどんと出るものをダイアログという。決まって画面の真ん中に出てきて移動できない。「確認」とか「取消」というスイッチのついたやつで、たいていスイッチが並んでいて、対話的にさまざまな設定を行うものだそうだ。…



も意味が広い，ハードウェアすべてから，ときには人間までコンピュータ資源のひとつとして扱うことになる。それではあまりにもあいまいなので，資源とはいわずにリソースという。

リソースというのは，ソフトウェア資源の一種，さらに狭義に，アプリケーションが扱うデータだと思って構わない。さらに正確にいうと，ここでいうリソースとは固有名詞である。リソースという名前の形式のデータがあって，リソース・ファイルというファイルに入っているのだ。その中身は音であり，フォントデータであり，グラフィックデータであり，メッセージであったりする。アプリケーション・プログラムとリソースを区別することにより，プログラムのデータ部分が独立するのである。

マネージャがたくさん

SX-WINDOWはウィンドウシステムである。ウィンドウシステムといってもSX-WINDOW全部をひっくるめてウィンドウシステムといっているのであって，ひっくるめてしまうとよくわからない。

SX WINDOWの基礎として，SX WINDOWを2つに分けて考えよう。

ひとつは実際に制御するシステムの部分である。シェルであるSXWIN.Xと，デバイスドライバであるFSX.Xがそうだ。COMMAND.Xをコマンドシェルというのと同様，SXWIN.XはSXシェルという。FSX.Xが実際にいろいろ資源を管理したりコントロールしたりする。この2つが働いて，目に見えるSX WINDOWというものができあがる。エンドユーザーはこのシェルとしての顔しか目に触れないわけだ。

ちなみに，シェルというのは昭和シェル石油のシェルと同じ。貝の殻のことであり，殻のようにOSを包んでいるのでシェルと呼ばれるようになった(らしい)。

もうひとつは開発支援としてのSX WINDOWシステムである。これはCやアセンブラのライブラリとして提供され，プログラムする人は自分で余計な世話はせず，みんなこれらに任せなければならない。下手に自分でやると，行儀が悪いといわれてしまう。

さて，こういってしまうと単純だが，SX WINDOWはウィンドウシステムとしてさまざまな資源を管理するため，動作するのも大変である。コマンド環境のときは最低限の入出力とBIOSを提供しておいて，あとはアプリケーションに任せていたので，システムは少ないメモリで動作した。ウィンドウ環境ともなると，その環境で使われるすべてについて管理せねばならないので，システム自身がたくさんのメモリを必要とし，さらに「グラフィカル」なユーザーインタフェイス用にもメモリを使う。そして，要2Mバイトとなってしまう。ご苦労さんである。

そういった管理をするために，SX WINDOWはたくさんのマネージャを持っている。SXシェルはあくまでもシェルであり，働くのはこのマネージャたちなのだ。プログラマがプログラム中でSX WINDOW用ルーチンをコールすると，コールされたルーチンたちが実行時にマネージャをこき使うのだ。マネージャはたくさんいて，それぞれが自分の大事な仕事を抱えている。

彼らはどこにいるか，FSX.Xの中だ。

マネージャというのは管理者という意味。SX WINDOWの持っているマネージャは次のとおりだ。

なお，マネージャの名前が「〜マン」となっていて，少し恥ずかしい気もするが，メモリマンと一般的な意味でのメモリマネージャを区別する必要がある場合などを考えると，まあ，しかたないと納得しておこう。そのうち，もっと凄い「グラフマンタロウ」とか「ダイアログマンジョーニアス」とか出てくるかもしれないし。



MEMORY　メモリマン

メモリマネージャのこと。メモリ全部の管理をする。新しくアプリケーションが起動するとメモリを確保し，クローズすると開放する。そんな大事な仕事だ。メモリマンは日夜陰で働いているのだ。

EXCEPTION　エクセプションマン

エクセプションマネージャのこと。エクセプションというのは割り込み。V-DISP(垂直帰線)割り込みをサポートする。アプリケーションで使うことはあまりないだろう。陰の下グループの一員。

MOUSE　マウスマン

マウスマネージャ。マウスの座標計算をしたりする。マウスカーソルを表示するときにはアニメーションマンを呼んでやってもらう。マウスがクリックされると，イベントマンを呼んで後始末をしてもらう。それが仕事の彼である。

ANIMATION　アニメーションマン

アニメーションマネージャ。アニメーションといってもCGAしてくれるわけではないので誤解しないように，マウスカーソルのアニメーションを担当するやつだ。マウスカーソルのアニメーションはV-DISP割り込みを使うので，エクセプションマンと仲好しである。

KEYBOARD　キーボードマン

キーボードマネージャ。キーボード関係の制御をするのだが，特に彼の世話になることはないだろう。陰の下グループの一員だ。

KEY　キーマン

キーマネージャ。キーマンというほどの者ではない。キーが押されるとキーボードマンからそのデータを受け取り，キーが押されたイベントを発生するのが仕事だ。彼も陰の下グループの一員。

EVENT　イベントマン

イベントマネージャ。別に地方イベント担当の電通社員ではない。イベントというのはマウスがクリックしたりキーが押されたりウィンドウが開いたり動いたりといった出来事全般のこと。実のところ，タスクマンの下働きをしている。

RESOURCE　リソースマン

リソースマネージャ。リソースというデータを管理する人。プログラムで扱うデータをリソースってことにして，彼にまかせて，管理してしまおう。

GRAPHIC　グラフマン

グラフィックマネージャ。グラフマンはインドはバッモン思想の中心概念。グラフマンはSX WINDOWにおける描画の中心。SXシェルの外からでも使える。グラフィックといっても，テキスト画面にも描画できる。

WINDOW　ウィンドウマン

ウィンドウマネージャ。ウィンドウ作成や描画や消去を管理する。お世話になります。

MENU　メニューマン

メニューマネージャ。メニューというのはお馴染みポップアップメニューのこと。必要なデータを彼に渡してやると，ポップアップメニューしてくれる。

CONTROL　コントロールマン

コントロールマネージャ。コントロールというのは，ウィンドウをコントロールするアイテムのこと。具体的には，ボタンとかスクロールバーとか，そういったものだ。

DIALOG　ダイアログマン

ダイアログマネージャ。ダイアログというのは対話ボックス。ウィンドウのように動かしたり閉じたりできず，とにかく指示に従った操作をしない限り画面に居座ったままのやつのこと。たとえば，ディスクコピーやフォーマット時のサーキットの絵はダイアログだ。

TEXT　テキストマン

テキストマネージャ，まあ，テキスト入出力をやってくれるものだ。

TASK　タスクマン

タスクマネージャ。メモリマンと並んで，SXシステムの基幹を担う大御所。マルチタスクの要。たいていのアプリケーションは彼の下で動く。

*　　*　　*

というわけだな。それぞれが機能分担して働き，プログラムを動かす。SX WINDOW上でプログラムが正しく動くのは，マネージャさんが働くからといって過言ではない。ライブラリにはこいつらを働かせるためのルーチンが入っているわけだ。非常にたくさんマネージャさんがいる。そのため，新しく覚えなければならないことも多い。マネージャさんはその分，働き者なのである。

OSに管理されていないプログラムに慣れた人はどうしてこんな面倒なことをしなければならないのかと思うかもしれない。が，面倒だからと構造体やポインタを使わずC言語を使う人はいない。最初は慣れにくい概念でも，それが高次元な処理を実現してくれるからだ。SX WINDOWのマネージャたちが提供するサービスはかなり高度なものがある。それは，慣れてしまえば，これまでのプログラミングがひどく野蛮なものに見えるかもしれないと思わせるほどのものである。

いまはまだ，膨大な資料に圧倒されていてもいい。さらに詳しくは以降の記事や付録ディスクに記載されているのでそちらをどうぞ。これから何カ月かにわたってSX WINDOWの特集が続く予定だ。少しずつものにしていってほしい。

MacintoshとSX-WINDOWの人にはいえない怪しい関係

「SX-WINDOWのシステムコール(SXコール)はAライン・エミュレータによって行われる」

これは，X68000のCPUであるMC68000の例外処理動作を利用したものだ。通常，CPUはメモリ上からプログラムを読み込んで解釈実行する。このとき，CPUの仕様に定められていない命令を実行しようとすると，CPUはプログラムが暴走した危険があるとして，あらかじめ定められたエラー処理プログラムを呼び出す。

68000では命令コードを16進数で表したときFとAで始まる命令を持っていない。ほかにも抜けている命令があるのだが，これらは特に未実装命令と呼ばれている(それ以外は不当命令)。これらの未実装命令はAで始まる命令とFで始まる命令のときで呼び出すルーチンが異なる。CPUがそう設計されているのだ。

これを利用すると，処理ルーチン側でさらに分岐する。Aで始まる命令それぞれに対して違う処理ルーチンを割り当てることができる。つまり，ユーザー定義の命令が簡単に追加できるのが，これがAライン・エミュレータと呼ばれる機能である。すでに同様のFライン・エミュレータはX68000のDOSコール，浮動小数点演算コールなどで使用されている。

SX-WINDOWやMacintoshのToolboxは，このAライン・エミュレータを使っており，システムコールも非常に似ている部分がある。そこで編集部でSXコールと似ているMacintoshのコール名を調べてみた。Macintoshのプログラムを移植する際には役立つだろう。

ここに挙げたものはまだ一部にすぎない。さらに，完全に同じ機能とは限らないことに注意してほしい。



システムのしくみを探る

ウィンドウプログラミングへの道

Murata Toshiyuki　村田 敏幸

アプリケーションのないウィンドウシステムなんてとお嘆きの貴方、ユーザーだったら待っていないで自分でもなんとかしましょうよ。まずはその第1歩、基礎概念から、ウィンドウを出して文字を表示するまでを一気に走ってみてください。

発表以来、高いところから「ウィンドウシステムだぜ、えっへん」とかいいながら、もったいぶって遠いビジュアルシェルになっていたSX WINDOWが、ようやくアマチュアプログラマにも手を出せそうなところに降りてきました。お待ちかね、技術資料公開です。今月号の付録ディスクに、ドキュメント（以下、嘘を承知で"簡易マニュアル"と呼んでしまいます）、開発関連ツール、サンプルからなる開発キットが収録されていますから、まずは解凍して、その全貌に目を通してください。若干ショックが和らいで、興味がむくむくわいてきたところで、本文へどうぞ。アセンブリ言語レベルで「ウィンドウを開いて文字を表示するプログラム」を作ることを目標に、周辺から組織と概念の解説をだんだんとやります。

なお、これからする話のほとんどは簡易マニュアルを読めば書いてあることばかりです。いうまでもないことですが、マニュアルは無視せずにきちんと読みましょう（比較的読みやすく書かれていることでもありますし）。ところで、伏のマニュアルには"読むマニュアル"と"引くマニュアル"の2種類があるのはご存じですね。付録ディスク中のドキュメントファイルでは、～.DOCが読むマニュアル、～.REF、～.LSTが引くマニュアルです。

参考書

最初に市販の参考書の紹介からまいりましょう。とはいっても、SX WINDOWの解説書なんかまだどこにもありません。Macintoshの本を参考にしてしまおうという話なのです。噂されていたように、そして、解析してみた方ならすでにおわかりのように、SX-WINDOWはかなりの部分、MacintoshのToolboxをパクって、意識して作られています。グラフマネージャはQuickDrawに生き写し、テキストマネージャはTextEditのそっくりさん、メモリ、イベント、ウィンドウ、コントロール、ダイアログの各マネージャも完全コンパチ寸前の下位コンパチです。各ファンクションコールはほとんど1対1に対応し、プログラム中では記号定数で書き換えられるマジックナンバーまでわざわざ同じ値に揃えてあります。

あんまり似ているので、ついつい知的所有権のことが気になってしまいますが、もちろん、SX WINDOWのコード自体はシャープのオリジナルで、もしAppleに訴えられても勝てるだけの証拠が保管されているはずです。私たちはそんなことはまったく心配せずに、「シャープさんが私たちの便宜のためにMacintoshに似せてくれたのだ」と思って、利用できるものは利用しプログラミングに励みましょう。

■参考書を利用するうえでの予備知識

Macintoshの解説書を利用するときには、どこまでがSX WINDOWと同じで、どこから違うのかを見極めながら読まなければなりません。MS-DOSや8086用のCの本を、Human68kやXCプログラミングの参考にするときと同じ脳味噌の使い方が必要です。簡単に要点を挙げておきます。

1) 概念は共通
2) 用語はほとんど同じ（異なるものについてはあとでちょこちょこと触れます）
3) ファンクションコール名、コール番号はぜんぜん違う（ちょっと痛い）

また、Macintoshの主力開発言語はPascalで（最近はCも広く使われているようですが）、解説書もPascal主体で書かれていますから、Pascalの知識もあったほうがよいでしょう。もっとも、Cを知っていれば

```
Integer      →short
Longint      →long
Nil          →NULL
Boolean      →enum { FALSE,
                     TRUE, }
Record       →struct
Function foo(~):Integer
             →short foo(~)
Procedure proc(~)
             →void proc(~)
```

などなどとフィーリングで読み替えればなんとかなります、きっと。

●Inside Macintosh

Macintoshの資料といったら、なんといっても、Toolboxの開発者らの手による標準マニュアル『Inside Macintosh（Addison Wesley）』です。日本語版はトッパンからVolume I&II、III&IV、Vの3巻で発売されています。ぶ厚い百科事典のような本です。それぞれ20,085円、12,875円、13,900円（税込み定価）ですから、合計するとC compiler PRO-68K Ver.2.0の定価を越えることになります。もっとも、SX-WINDOW上のプログラミングの参考にするだけであれば、I&IIだけがあればよいでしょう。III&IVはMacintoshのハードの話＋I～IIIのサマリー総集編＋いわゆるFat Mac以降の拡張部分という構成ですし、VはMacII以降の拡張部分の補足解説にすぎず、Color QuickDrawについて書かれた数ページが僅かに関係あるかな？　という程度です。本当のところはIIもメモリマネージャの部分以外は役には立たないようですが、IとIIで1巻ですからしかたがないですね（それとも旧版に手を出しますか？）。

さすがに内容は充実しています。ほかの解説書では省略されている深い部分に関しては、この本に頼るよりありません。マニュアル臭をふんぷんさせ、サンプルプログラムはおろか、コール例もほとんどないのが珠に傷ですが、豊富な図版が理解を助けてくれるでしょう。また、各章末のサマリー部分には必ずPascalでの使い方と、アセンブリ言語レベルでの使い方の両方が揃っており、リファレンスとしても申し分ありません（本の厚さのせいで引きにくいのは確かですが）。ただ、文中の固有名詞はともかく、サマリー中のコメントまで英語のままですので、「どこが日本語版なんだ！」という憤りを感じることはあるかもしれませんね。

■マッキントッシュの道具箱

もう少しお手軽な本では『マッキントッシュの道具箱（パーソナルメディア）』があります。Vol.I、IIの2巻で各4,944円（税込み定価）です。Vol.IがメモリマネージャとQuickDrawなどの基本部分、IIがウィンドウ関係、という構成になっています。著者は『Inside Macintosh』にも関わった人のようで、その点、信頼できます。各章末のリファレンスも適度によくまとまっていて、とくにワンポイントアドバイス的なノート部分は簡潔に要点を示してくれています。コール例＋α程度のサンプルプログラムも多数用意されており、とくにVol.IIの巻末にはテキストエディタの全ソースも載っています。ただし、ソース中のコメントは訳されていません。あ、翻訳といえば、私はこの本の訳があまり気に入っていないということをつけ加えておきましょう。

『マッキントッシュの道具箱』には『続マッキントッシュの道具箱』という続編もあります。もっとも、こちらは著者・訳者とも別人で、本の性格も異なります。Macintosh上でプログラムを作る際の注意点やデバッグ技法について述べられた読み物です。資料性の高い本ではありませんが、メモリ管理絡みの話などは参考になるでしょう。SX-WINDOWには関係ない部分も、「あっちのプログラマは、こうやってデバッグしているんだな（同じだな、違うな、いいな、なにそれ）」と考えながら読めば、それなりに楽しめます。

●インサイドマック徹底ガイド

さらに手軽なところでは『インサイドマック徹底ガイド（BNN）』があります。上下巻、各3,300円（税込み定価）です。価格も手ごろで、内容もよくまとまっており、なかなかのお勧め品といえましょう。上巻だけでも十分役に立ちます。というより、下巻はほとんどのページをMacintoshのOSに割いているため、あまりSX-WINDOWには関係ないのです。

「How To Use Inside Mac」という副題が示すように、内容は『Inside Macintosh』の道筋を補うためのガイドといった感じになっています（そういいきってしまうのは失礼かもしれませんが）。『Inside Macintosh』から必要最小限の部分を抜き出してきたような作りです。その分、エッセンスが詰まっているともいえます。部分的に抄訳っぽい部分も見られるものの、基本的には純国産ですから、訳の悪さに煩わされることもありません。小中編のサンプルも適宜にちりばめられており、上巻の巻末には大きなサンプル（テキストエディタ）の全ソースも載っています。

●雑誌記事

雑誌記事では「インターフェース」誌1988年12月号の特集"Macintoshの技術徹底解剖"がかなり参考になりました。サンプルプログラム（またしてもテキストエディタ）がCで書いてあるのが見どころです。

また、私は雑誌の色が強いなのでまじめに読んだことはありませんが、Mac専門各誌にも目を通してみれば、なにか得るものがあるでしょう。

SX-WINDOWシステムの概要

プログラミングに入る前に、周辺の話題を落穂拾い的に取り上げてみます。

●バージョン

SX-WINDOWにはEXPERT II/PRO IIに添付されたVer.1.00、SUPER-HDに添付されたVer.1.01、そして、パッケージ版のVer.1.02があります。これだけ頻繁にバージョンアップしたところをみると、ぼろぼろとバグが出たのでしょう（と、知らないふりをする）。エンバグしてなければいいですね（と、知らないふりをする）。Ver.1.01以前の登録ユーザーに対しては無償でバージョンアップサービスが行われたようですから、すでに皆さんの手元にはVer.1.02があることと思いますが、もしまだ古いバージョンを使っているようであれば、早急に手を打つことをお勧めします。バージョンアップの過程では、たんなるバグとりだけではなく、処理速度の向上や、機能の追加もあわせて行われているからです。

●システムとしての位置づけ

SX-WINDOWシステムの本体はFSX.Xという1本のプログラムで供給されています。FSX.XはCONFIG.SYSにより組み込まれるデバイスドライバとして、また、起動後にコマンドラインから実行する常駐プログラムとしてメモリに居座り、Human68kに憑りついて、ウィンドウシステムとしての機能を付加します。見掛けは、ほかのデバイスドライバの類、たとえばFLOATn.XがHuman68kに浮動小数点演算機能をつけくわえるのに似ています。しかし、FLOATn.Xがコブのように、Human68kにくっついたのに対し、SX-WINDOWはHuman68kをすっぽり包み、さらにはIOCSやハードの一部にまで触手を伸ばして管理下に置き、全体をひとつのシステムにまとめ上げるのです。

シェルであるSXWIN.X（以下、簡易マニュアルに従いSXシェルと呼びます）も、SX-WINDOWシステム上の1アプリケーションにすぎません。ちょうど、VS.XがHuman68kのアプリケーションのひとつだったようにです。その気になれば、べつのシェルを作って差し替えることも可能ですし、SXシェルを使わずに単独で動くプログラムを作ることもできます。

ところで、コマンドラインから組み込んだFSX.Xは、もう一度FSX.Xを起動することで常駐解除することができます。常駐プログラムは/rスイッチによって切り離すように作るのがふつうですが、FSX.Xはスイッチなしなのですね。私は意地が悪いので、それが何を意味するのか知りつつ、SXシェル上からFSX.Xをダブルクリックして遊ぶのでした（こういう小さな部分でシステムの信頼性を計るのは筋違いだと思いますか？）。

■SXコール

SX-WINDOWのファンクションコール（以下、簡易マニュアルに従いSXコールと呼びます）はA系列未実装命令を使って呼び出します。Human68kのDOSコールやFLOATn.XのファンクションコールはF系列の未実装命令を使って呼び出すわけですが、$Fxxx_H$の代わりに$Axxx_H$を使うこと以外はSXコールも似たようなものです。引数もDOSコール同様、スタックに積んで渡します。

SXコールは公開されているものだけでも500個弱あります。Human68kのDOSコールが100個ちょっと、IOCSコールが200個程度ということを考えると、とんでもない数です。しかも、公開されていないものがこのほかにもまだあり、ざっと数えてみただけでも100個以上のSXコールが未公開のまま眠っています。なかには、バグのため使えないものとか、"コーヒーのないクリープ（誤植じゃないぞ）"のごとき無意味なものもあるようですが、せめて既存のデスクアクセサリなどで使用しているSXコールはすべて公開してほしかったところです。

■イベントドリブン

イベントドリブン（event driven：イベント駆動）とは、プログラムがなんらかの出来事（＝イベントの発生）をきっかけとして駆動される様子を表す言葉です。イベントが発生するのをじっと待ち、発生したらそのイベントの種類に応じた処理を行い、また次のイベントを待つ、そのように作られたプログラムのことをイベントドリブン

型のプログラムと呼んだりもします。

ここでいうイベントとは、プログラムに対する外部からの働きかけの総称です。主にキー入力やマウスボタンのON/OFFといったユーザーのアクションを指しますが、もっと抽象的なもの、内部的なものもイベントの範疇に含みます。抽象的なイベントとしては"イベントがないことを表す"イベントとか、システムからの合図の類などが考えられます。プログラムにとってはユーザーのアクションもシステムからの合図も外部からの働きかけには違いないわけです。

イベントドリブンの考え方は、複数のイベントが非同期的に起こるような状況でとくに有用になってきます。

第1に、プログラムの構造がすっきりします。イベントドリブン型プログラムのメインルーチンは、

1) イベントを待ち受ける
2) イベントの種類に応じて処理を振り分ける
3) 1)に戻る

という、非常にシンプルなループの形にまとめることができるのです。しかも、イベントドリブン型プログラムを前提にしたシステムでは、1)のイベントを取得する部分がシステムコールとして提供されるのがふつうですから（もちろんSX-WINDOWもそうです）、プログラム側の負担はさらに軽くなります。個別にキースキャンしたり、マウスのボタンの状態をチェックしたりといったことをしなくても、システムに漠然とイベントを要求すれば、"このキーが押された"とか"マウスの左ボタンが押された"といったイベントが返ってくるわけです。

プログラムをイベントドリブン型にする第2のメリットは、ユーザーにとってのものです。イベントドリブン型プログラムでは、基本的にすべてのイベントが対等に扱われます。イベントに対応する処理ルーチンはおのおの独立していますから、呼び出す順序＝イベントの発生順序＝ユーザーのオペレーションの順序にも制限がありません。

ウィンドウシステム上では、マウスボタンが押されたり離されたり、キーが押されたりといったイベントがユーザーの気の向くままランダムに発生します。プログラム側はそれに備えなければなりませんし、ユーザー側もそれを期待します。となると、ウィンドウシステムにイベントドリブンという組み合わせは、プログラムを作る側にも使う側にもなかなかおいしい選択といえるのではないでしょうか。

■イベントドリブン型の擬似マルチタスク

イベントドリブンは直接マルチタスクに結びつく概念ではありません。現にMacintoshのシステム（ただし、MultiFinderなしの標準状態）はシングルタスクでありながら、各プログラムはイベントドリブン型をしています。両者をつなぎあわせるのには多少の小細工が必要になってきます。

ところで、マルチタスクという言葉は説明する必要ありませんね？　複数のタスク（task　仕事、処理の単位）が並行動作している（ように見える）状態を指す言葉です。プロセッサがひとつしかないふつうのコンピュータでは、タスクを少し動かしては止め、また別のタスクを動かして、というようにシステムが調停することでマルチタスクが実現されます。ちなみに、タスクに類した単語に、ジョブ（job）、プロセス（process）、多少ニュアンスは違いますがスレッド（thread）などがあります。英語ではそれぞれ"仕事"とか"一連の処理"といった程度のぶっきらぼうな言葉です。根本的な意味の違いはないと考えてよいでしょう。ただ、システムによって、仕事・処理の単位にいろいろな呼び名をつけて区別しているのです。余談でした。

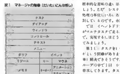
では、イベントドリブン型のプログラムがマルチタスクで走るためにはなにが必要で、実現するにはどうしたらよいか、考えてみましょう。

第1に、タスクをいつどこで切り換えるかという問題があります。これは比較的簡単に解決できそうです。各タスクはメインループの中で1回、システムにイベントを要求してきます。タスク側から見れば、単にシステムコールを発行したというだけのことですが、システム側から見れば、自分に制御が戻ってきたことになります。ここにタスクを切り換える隙があります。タスクAから戻ってきたときに、タスクBにイベントを返し、タスクBから戻ってきたらタスクAにイベントを返す、という具合です。

第2に、十分短い間隔ごとに、各タスクに制御を渡してやらなければなりません。これは、イベントが発生するまでじっと待つイベントドリブン型プログラムとは相性が悪いようにみえます。しかし、逆にいうと、イベントが頻繁に発生しさえすれば、タスクに制御が移る回数も多くなるわけですから、意味のない架空のイベントを作り上げるという案が浮かびます。ヌルイベント、つまり"イベントがないことを表すイベント"です。イベントが発生していないときには、システムは適当な間隔で、このイベントを返し、眠っていたタスクを起こします。タスクはしばらく走って、また、次のイベントを要求するわけです。

そして、この2つの案を組み合わせると、SX-WINDOWのイベントドリブン型の擬似マルチタスクになります。"擬似"がついているのは、タスク側の協力がないとマルチタスクにならない、という意味にとりましょう。制御を渡したが最後、システムにイベントを要求しないプログラムがひとつあるだけで、マルチタスキングはぴったり止まってしまいます。そこまで酷くなくても、のろまなタスクがいれば、システム全体の速度ががたんと落ちます。システム側には、制御を強制的に自分に引き戻す手段がないのです。

図1　マネージャの階層（だいたいこんな感じ）



SX-WINDOWの各マネージャ

マネージャとは、なにか特定の分野を一括して管理するプログラム/ルーチン集をいいます。SX-WINDOWシステムもいくつかのマネージャから構成されています。もっとも、プログラム自体はFSX.Xひとつにまとめられているのはご承知のとおりです。マネージャという呼び名は論理的な区分けにすぎません。

SX-WINDOWシステムには、15のマネージャがあります。マネージャには低水準なものから高水準なものまで、数段階のランクがつけられています（部分的には曖昧なところもありますが）。いちおう図1に各マネージャの階層を示しておきます。図中上にいくほど高レベル・抽象的になり、同時に下位のマネージャに依存する度合いが増す、と思ってください。

では、以下、各マネージャの働きを簡単に紹介しましょう。重要なものについては、あとでまた掘り返ります。

■メモリマネージャ

メモリの総括的な管理を行います。上位のマネージャやアプリケーションプログラムは、メモリが必要になったらメモリマネージャにメモリを要求し、確保してから、使うことになります。

■エクセプションマネージャ

非常に低水準な、マネージャと呼ぶのもおこがましいようなマネージャです。ひとつのエクセプション（exception　例外）に対して、複数の処理ルーチンをぶら下げることができるよう細工します。もっとも、いまのところは、垂直帰線信号（V-DISP）による割り込み専用です。SX-WINDOWはシステム時間を計るタイマに垂直帰線信号を使っています。

●マウスマネージャ

マウスの移動、マウスカーソルの表示を担当します。マウスデータを受信するところまではIOCS同様にROMの割り込みルーチンを流用し、受信したデータだけを盗んできて細工を加えているようです。これにより、マウスの移動量や、移動方向の左右をソフト的に変更することができるようになっています。

●アニメーションマネージャ

垂直帰線信号による割り込みを用いてマウスカーソルのアニメーションを行うだけのマネージャです。簡易マニュアルによればエクセプションマネージャの一部だそうです（さもありなん）。

●キーボードマネージャ

キーボードからキーデータを得て、キーバッファに溜め込みます。やっていることはROMのキー入力割り込みルーチンとさほど変わりません。また、提供されるサービスもIOCSのキー入力関係コール程度のものですが、XF1～XF5をシフトキーとみなすといった細工が施されているのが面白いところです。

●キーマネージャ

キーボードマネージャから生のキーデータを受け取り、CONデバイスを通してASCIIコードに変換します。これは日本語入力FEPを有効にするための処置です。

●イベントマネージャ

イベントの管理をします。イベントマネージャは、一部ウィンドウ関係のイベントも扱いますが、マウス、キー関係マネージャの元締めだと思っていればよいでしょう。マウスボタンの押し下げやキー入力をイベントとして内部のキュー（待ち行列）にため込みます。ため込んだイベントはのちにアプリケーションや上位マネージャから要求があったときに、ひとつずつキューから取り出されることになります。SXシェル上でマウスボタンを闇雲に何度も押すと、押した回数だけメニューがちかちか点滅するといった症状が見られますが、これはマウスボタンのON/OFFがイベントとしてキューにたまった結果です。

なお、イベントマネージャにイベントを要求したときにはタスクの切り替えは行われません。タスクの切り替えが起きるのは、タスクマネージャにイベントを要求したときです。

タスクマネージャはイベントマネージャからイベントを取得し、タスクマネージャ独自のイベントとともにアプリケーションプログラムに渡します。逆にいうと、タスクが切り替わってほしくない一時的なマウスボタンのチェック/キー入力のときには、タスクマネージャを通さずに、イベントマネージャに要求すればよいのです。

●リソースマネージャ

リソース（resource）の管理をします。リソースというのは……、あとでねっとり説明しますね。

●グラフマネージャ

SX-WINDOWシステム上のほとんどの画面描画を担当する、高級なテキスト/グラフィック描画ルーチン集です。MacintoshのQuickDrawに相当します。多くの場合、テキスト画面とグラフィック画面のどちらに対しても同様の描画が行えます。現行のSXシェルでは画面モードの関係でグラフィック画面は16色固定ですが、グラフマネージャ自体は、きちんと256色、65536色モードもサポートしています。65536色モードではColor QuickDraw並の色の混ぜ合わせもできるようです。

なお、Macintoshでいうフォントマネージャは、機能を大幅に縮小した形でグラフマネージャに併合されています。

●メニューマネージャ

ポップアップメニューの管理をします。Macintoshに比べるとかなり小規模なマネージャになっており、マネージャ自身を初期化する、リソースファイルからメニューを読み込んで使える状態にする、マウスの動きを追跡してメニューを選択する、の3つのファンクションコールしか用意されていません。実際の使用にあたっては、多少周辺ライブラリを整備してやる必要がありそうに思います。

●テキストマネージャ

テキストマネージャは、MacintoshのTextEditに相当し、基本的なテキスト編集機能を提供します。先ほど紹介したMacintoshの本でサンプルがみなテキストエディタだったのは、TextEditの存在ゆえです。SX-WINDOWのノート.Xも多分テキストマネージャを使って作られていると思われます（確認はしていません）。また、テキストマネージャはファイル名の入力などに広く利用されています（簡易マニュアルによると、そちらが本職だそうです）。

●コントロールマネージャ

コントロールというのは、ウィンドウ上に配置されたスクロールバーや各種ボタンの総称です。コントロールマネージャはそれらの制御をします。

●ウィンドウマネージャ

ウィンドウを管理・制御します。

●ダイアログマネージャ

ダイアログ（dialog box）は"対話ボックス"とか、"対話型ウィンドウ"などと訳されます。ユーザーに指示や確認を求めたりといった目的に使われる対話用のウィンドウのことです。ダイアログマネージャはこのダイアログの管理、運営をします。ダイアログ上には多くの場合、さまざまなコントロール類が置かれますが、ダイアログマネージャはこのコントロールの面倒もみてくれます。

なお、ダイアログはウィンドウの特別な形ということもあり、ダイアログに対してもウィンドウマネージャのファンクションを適用できます（そのはずです）。

●タスクマネージャ

タスクの管理をするSX-WINDOWシステムの総元締めです。Macintoshでいうスクラップマネージャやファイルマネージャの機能も包含しています。

メモリ関連

そろそろプログラミングの準備に入りましょう。まずはメモリマネージャによるメモリ管理関連を押さえておきます。

メモリマネージャはメモリをヒープ（heap）の形で管理します。ヒープは1個以上のメモリブロックからなり、メモリ関係要求があることに、メモリマネージャは要求バイト数を満たす空きメモリブロックを検出し、その一部ないしはすべてを確保します。なお、メモリマネージャは複数のヒープを管理することもでき、現在使用中のヒープをカレントヒープと呼びます（複

数のヒープを使う場面はあまりないと思われますが)。

メモリブロックにはいくつかの属性(種類)があります。まず、少なくとも、各ブロックは割り当て済み(allocated block)か未割り当て(free block)かのどちらかです。さらに、割り当て済みのブロックは図2のようないくつかの種類に分類されます。

リロケータブル(再配置可能:relocatable)なブロックは、メモリマネージャによってヒープ内を移動させられる可能性のあるブロックです。メモリの確保・解放を繰り返すことでヒープは図3のa)に示すように細切れに細分化される場合があります。一度に確保できるメモリブロックの最大の大きさは、最大の空きメモリブロックの大きさに制限されますから、空きメモリの総量が十分あっても必要なだけのメモリブロックが確保できないという事態が生じることになります。そこで、メモリマネージャは図3のb)のようにリロケータブルなメモリブロックを動かし、詰めて、要求されただけの大きさのメモリブロックを作ろうとするわけです。このメモリブロックを詰める操作をコンパクション(compaction:コンパクト化)といいます。

さて、コンパクションの発生によって、リロケータブルなブロックは勝手にどこかに動かされます。その結果、図4のように、このブロックを指していたポインタは意味をなさなくなってしまいます。これではまともなアクセスはできませんね。というわけで、実際にはリロケータブルブロックはポインタではなく、ハンドル(handle)によって参照されます。

ハンドルというのは"何かを間接的に表現する代名詞"を意味する用語ですが、ここでいうハンドルは"リロケータブルブロックを指すポインタ(=マスタポインタ:master pointer)を指すポインタ"のことです。マスタポインタはヒープ上の固定領域にいくつもまとめて置かれ、個々のマスタポインタはリロケータブルブロックを指しています。そして、コンパクションが発生したときには、マスタポインタも同時に書き換えられ、つねに正しくリロケータブルブロックを指すよう保たれます。

1) マスタポインタは動かない

2) マスタポインタはつねにリロケータブルブロックを指している

の2点から、ハンドル(マスタポインタへのポインタですよ)による2重間接を適用することにより、いつでもリロケータブルブロックへのアクセスが可能となるわけです。図4にそのイメージを示します。

対して、ノンリロケータブル(non relocatable:再配置不可)なメモリブロックは、ヒープ上に固定され、動くことのないメモリブロックです。ノンリロケータブルなブロックは、ポインタによって参照することが可能であり、ハンドルを介するリロケータブルブロックよりはアクセス速度(手間)の点で有利ではあります。が、図5に示すようにノンリロケータブルブロックはコンパクションの妨げとなりますので(メモリマネージャはほかのブロックをまたいでブロックを動かすことはありません)、効率的にメモリを扱おうと思ったら、なるべくノンリロケータブルブロックは使用しないようにすべきです。

さて、リロケータブルブロックは、もし望むのであれば、コンパクションの対象から外すこと、つまり、その位置を固定することができます。位置を固定してしまえばマスタポインタのコピーを作ってアクセス速度を稼ぐことが可能になるわけです。ただし、固定された(locked)メモリブロックは、やはりコンパクションの妨げとなりますから、アクセスが済んだら忘れずに移動可能(unlocked)に戻す必要があります。なお、リロケータブルブロックの初期状態はunlockedです。

またさて、ときには、コンパクト化してもなお、メモリが足りない場合があります。それでも、メモリマネージャはあきらめません。リロケータブルで、かつ、固定もされていないブロックの中から、不用なものを削除し始めます。これをパージ(purge)といいます。もちろん、手当たり次第にメモリブロックを捨てられてはたまったものではありませんね。メモリマネージャがパージの対象にするのは、パージ可能(purgeable)という属性がついたメモリブロックだけです。リロケータブルブロックの初期状態はパージ不可(unpurgeable)になっていますから、アプリケーション側で意図的にパージ可能にしないかぎり、勝手に消されたりはしません。

プログラムの実行に必要なデータなどをディスクからメモリに読み込んで使う場合を考えましょう。そのデータは、使用される瞬間だけメモリ上に存在すればよいわけですから、極端な話、使いたいときに毎回ロードしてもよいことになります。が、それではロード時間が馬鹿になりません。といって、使わないときにもメモリ上にデータが居座っているとそれだけメモリを圧迫します。こういうときには、そのデータを置くメモリブロックをパージ可能にしておくのが好ましいといえます。データを使用したいときに、パージされてしまっているかどうか調べ、まだ生きていればそのまま使い、さもなくばメモリを再確保してディスクからデータを読み込み直すというわけです。

なお、リロケータブルブロックがパージされたときには、マスタポインタは0に書き換えられます。マスタポインタが0かどうかで、パージされたかどうかがわかるのです。値が0のマスタポインタを指す空のハンドル(empty handle)は、専用のファンクションコールを使って再割り当てして使います。

リソース

●SX-WINDOWにおけるリソース

簡易マニュアル中、RESOURCE.DOCは、"SX-WINDOWでいうリソースはMacintoshのそれとはまったく別物である"という断り書きから始まっています。確かに、ファイルシステム自体が大きく違うということもあり、Macintoshとはリソースの実現・管理方法が違います。ですが、機能的には非常に忠実にMacintoshのリソースをパク、いえ、模倣しているところもあり、使用のしかたなども似たようなものです。管理方法だけを指して"まったく別物"というのはちょっと言いすぎです。では、どこが違うのでしょう。

つらつら考えますに、たぶん、根本的なスタンスが違うのです。ポリシーの有無ともいえます。Macintoshのリソースはある意味でシステムの中核を成すのに対して、SX-WINDOWのそれは"主"ではなく"従"なのですね。積極的にアプリケーションにリソースを利用してもらおうという姿勢が最初からありません。そう考えていくと、アイコンがエディットできないといった点にも合点がいきます。

●リソースとは

さて、そもそもリソースとはなんなのかまだ説明していませんでした。簡易マニュアルでは、"ファイル名の代わりにリソースタイプとリソースIDによって識別される特殊な形式のファイル"といっています。SX-WINDOWのリソースは、ちょうどアーカイブファイルかライブラリファイルのように、拡張子がLBのファイルにいくつも詰めて格納されますから、そのような言い方もできるでしょう。でも、なんかピンときませんね。

多少独断じりで翻訳しますと、プログラムの実行に必要なデータや補助的な実行コードを、メインプログラム(のファイル)から分離したもの、それがリソースです。いわゆる、データファイルや、オーバーレイされるサブモジュール、ダイナミックリンクされるライブラリ的なモジュールなどは、みんなリソースとなりえます。

そうするとどのようなメリットがあるのでしょう。まず、プログラムを作る側からいえば、データなどの細かな修正や追加時にもプログラム本体を再アセンブル(再コンパイル)する必要がなくなります。これは、場合によっては、ユーザーレベルでプログラムの部分的なカスタマイズができることを意味します。また、分離されたリソースは必要なら複数のプログラムで共用することもできます。まさに、万歳なのです。

もっとも、プログラム本体とリソースを分離することによって管理が繁雑になるのもまた事実です。SX-WINDOWでは複数のリソースをひとつのリソースファイルにまとめておくことによって、辛うじてこの問題を回避しています(Xファイルを拡張するといった案は浮かばなかったのですか?)。ちなみに、Macintoshの場合は、ファイルシステムそのものがリソースを扱う前提で設計されています。簡単にいうと、"プログラムとリソースは物理的には分離されているが、論理的にはひとまとめ"です。

■リソースタイプとリソースID

リソースは、リソースタイプとリソースIDの2つの属性によって区別・識別されます。リソースタイプは4バイトからなり、通常、意味を持った(そして読める)4文字のASCIIコード列が使われます。たとえば、WDEFという名前で知られるリソースタイプの値は、W、D、E、Fそれぞれを ASCIIコードにして並べた

57444546$_H$

となるわけです。

リソースIDのほうは、同種のリソースタイプ内での識別番号で、−32768〜+32767の2バイト符号付き数で表されます。識別番号ですから、同種のリソースタイプ内で

図2 メモリブロックの種類

```
block
├─free block
└─allocated block
    ├─non-relocatable block
    └─relocatable block
        ├─locked block
        └─unlocked block
            ├─purgeable block
            └─unpurgeable block
```

図3 コンパクション

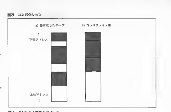


図4 ハンドルとマスタポインタ

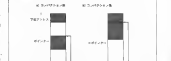
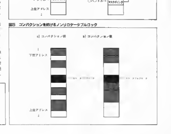


図5 コンパクションを妨げるノンリロケータブルブロック



は同じIDを複数のリソースに割り当てることはできません。また，負の値と0～127まではシステム予約で，ユーザーアプリケーションで使用するリソースには128～32767の範囲のみが使えることになっています。

ところでMacintoshでは，新しいリソースタイプ名が必要なときにはApple社にお伺いを立てて調整すると聞きます。SX-WINDOWではどうなるのでしょう。

■リソースの作成

リソースは，リソースエディタないしはリソースコンパイラで作成するものです。前者は，対話型のリソース作成・修正用のツール，後者は非対話型であらかじめリソースを簡易言語で書き表しておき，コンパイルすることでリソースを作り出すツールです。ですが，いまのところSX-WINDOWではそのどちらもサポートされておらず，リソースはアセンブラを使って作るしかありません。つまり，dc疑似命令でリソースの内部構造を再現するわけですね（個々のリソースの具体的な構造はここでは触れません。SXCALL.MACないしはSXDEF.Hを参照してください）。状況はかなり悲惨です。

そのためかどうか，リソースを積極的に利用しているのはシステム本体とSXシェルだけで，デスクアクセサリあたりになるとほとんどリソースを活用していません。唯一，コントロールパネルは，専用のリソースファイルCTRLPNL.LBを持ち，分離できるものはとことん分離した，なかなか模範的な作りになっています。

リソースエディタはともかく，リソースコンパイラぐらいなら，パワーユーザーの手の届く範囲だと思われます。公開ソフト市場に出回るのを待ちましょう（私はそれほど他力本願ではありませんけど）。

■リソースファイル管理ツール

リソースそのものを作るツールは未サポートながら，リソースファイルの管理ツールは今回ディスクに収録された開発キットの中に入っています。RLK.Xがそれです。どこにも解説がないようですが，コマンドラインから単に，

A>RLK

として起動すればヘルプが出ますから，すぐにでも使えるでしょう。アーカイバやライブラリアンと同じような使い勝手です。ただし，RLK.Xがサポートしているのは，

- リソースの更新・追加
- リソースの削除
- リソースの一覧表示

のみで，なぜか抽出ができません（解析師封じ？）。公開ソフトに期待しましょう。

使用上の注意を少々。当然のことですが，システムが使用しているリソースをむやみに削ったり改変したりしてはいけません。自分がなにをやろうとしているのか，その結果なにが起きるか理解したうえで利用してください。少なくとも，リソースファイルのバックアップはとっておくべきです。もっとも，現状ではビープ音を差し替えて遊ぶぐらいの使い道しかないかもしれませんね。例のぴんぽーん音はリソースタイプBEEP，IDは0として，SYSTEM.LB中にあります。

もう1点。RLK.Xは非対話型のDOSコマンドですが，実行時にはFSX.Xが組み込まれている必要があります。

●SX-WINDOWのリソースファイルの中身

SX-WINDOWには4本のリソースファイルがついてきます。ここで，簡単に中身を紹介しておきましょう。上述のRLK.Xを使えば一覧表示を見ることができますから，各自，のぞいてみてくださいね。

SYSTEM.LBには，システム本体（一部SXシェル）で使うリソースが収められています。ウィンドウ，コントロール，メニューの種類に応じた制御・描画を行う実行コード（定義関数といいます），ビープ音用のADPCMデータ，プリンタマネージャらしき実行コードと複数のプリンタに対応するためのドライバ，SXシェルの背景模様用のグラフスクリプト（あとで説明します）などです。

BUILTIN.LBはSXシェル上のビルトインコマンド関係のリソースファイルです。ディレクトリを表示する例のウィンドウや，メニューから起動されるそのほかのプログラムの実行コード，また，各メニューのデータなどが収められています。

ICON.LBはアイコン関係です。アイコンのパターンや，アイコンとファイルとの対応を示したリソースなどが入っています。

CTRLPNL.LBは，先ほども出てきたようにコントロール.X用のリソースファイルです。アイコンのパターン，ダイアログテンプレート（ダイアログを開くときの引数をリソースにしたもの），アイテムリスト（ダイアログ上のコントロールの種類，配置を示したリソースで，ダイアログテンプレートに結びついている），さらには，各サブパネルそれぞれ用の実行コードが収められています。

ついでといってはなんですが，表1にSX-WINDOWで使っているリソースタイプの一覧を示しておきます（これで全部かどうかは自信がありませんが）。

ところで，SX-WINDOW Ver 1.01までは，リソースファイル中に開発途中のものと思われる余計なリソースがたくさん残っていました。とくにSYSTEM.LBはゴミの宝庫で，ウィンドウ定義関数はダブっているわ，正常動作しないウィンドウ定義関数やコントロール定義関数があるわで，私たちのディスクスペースを無駄遣いしてくれていたのです。これらをばっさり削った結果，Ver 1.02のSYSTEM.LBはVer 1.01よりも30Kバイトほどファイルサイズが小さくなっています。まずは，めでたしめでたし。

表1　リソースタイプ

ALRT	アラートテンプレート（使えないかもしれない）
BEEP	ビープ音ADPCMデータ
BBPT	背景パターンリスト（16×16ドット 4プレーン 8種類）
CDEF	コントロール定義関数
CMDS	ビルトインコマンドとファイル/リソースとの対応リスト
CNTL	コントロールテンプレート
CODE	各種実行コード
DITL	ダイアログアイテムリスト
DLOG	ダイアログテンプレート
DME	ディレクトリウィンドウ用メニュー
ICN#	ファイルとアイコンの対応リスト
IMGS	イメージデータ
MDEF	メニュー定義関数
MENU	メニュー
PAT3	モノクロ4階調パターンデータ
PAT4	モノクロ4階調＋カラー3色パターンデータ
PICT	グラフスクリプト
PRTD	プリンタ機種別ドライバ
PRTM	プリンタマネージャ（？）
SHEV	SXシェル用データ（？）
SHME	SXシェル用メニュー
WDEF	ウィンドウ定義関数
WIND	ウィンドウテンプレート

●作成したリソースをどう使うか

簡易マニュアルには，既存のリソースファイルを利用する方法は記されています。リソースファイルをオープンしてから，リソースタイプとIDにより参照すればよいわけです。ところが，そこに至る道筋にはいっさい触れられていません（このあたりもユーザーにリソースを使わせまいとしない姿勢の現れ，と判断します）。すでに述べたように，リソースはリソースコンパイラやリソースエディタ，または，アセンブラを使って作ります。できたリソースはRLK.Xを使ってリソースファイルにまとめます。どのリソースファイルに？　それが問題です。開発側で想定する（あるいは想定している）方法が示されていないのです。

3案ほど考えました。ひとつは，既存のリソースファイルに収めてしまう方法です。SYSTEM.LBにでもいれておけば，SXシェル起動時に自動的にオープンされ，使える状態になります。ですが，システムのリソースと一緒にしてしまうのは，なんとなく気が引けます。

2案目は，プログラムそれぞれにリソースファイルを用意する方法です。コントロール.Xのように多くのリソースを使い，ほかのプログラムと共用するつもりもなければよい選択のように思えます。しかし，それほど多くのリソースを使わないのであれば，小さなリソースファイルがたくさんできる羽目となり，管理が大変になります。

で，3案目です。自作プログラム用のリソースファイルを1本（たとえば，USER.LBといった名前で）用意して，その中にいろいろなプログラムのリソースを詰め込むという案です。結局，これが一番自然な方法なのでしょう。自作のプログラムでは起動時にこのリソースファイルをオープンするという，自分なりの約束を作るわけです。この方法にも問題がないわけではありませんが，話が具体的になりすぎそうなので，突っ込むのはまたの機会にします。

グラフマネージャ

●基礎概念

グラフマネージャは，“グラフ”と呼ばれる描画環境に対して描画を行います。ほんとうは少し違うのですが，グラフ＝ウィンドウと考えてみてください。グラフマネージャは個々のウィンドウのどれかひとつを描画対象にするわけです（現在描画対象になっているグラフをカレントグラフと呼びます）。グラフは構造体様のデータブロックで，その中身は，ビットマップ（へのポインタ），描画色や，描画モードなどといった，描画環境に関する情報から成ります。

ここで，ビットマップはグラフに結びつけられた仮想画面の種類や大きさ，アドレスを保持する構造体です。グラフマネージャにとっては，このビットマップで示されるメモリにデータを書き込むこと＝描画となります。“仮想”画面といってしまいましたが，通常，ビットマップはVRAMそのものを指しており，その意味では実画面です。ただ，グラフマネージャにとっては，VRAMであろうが，メインメモリ上に取られた仮想画面であろうが，たいした違いはありません。

グラフマネージャはテキスト画面とグラフィック画面をほとんど区別することなく扱えるようにできています。X68000のテキストVRAMはいわゆる水平型，グラフィックVRAMは垂直型と，両者の構造は異なりますが，グラフマネージャはビットマップにテキスト画面用のものとグラフィック画面用のものの2種類を用意することで，その違いを吸収します。ビットマップを差し替えるだけで，テキスト画面に対してもグラフィック画面に対しても，同様の描画が行えるのです。

●座標系

グラフマネージャは2種類の座標系を使い分けます。ひとつはグローバル座標であり，これは実画面上の絶対的な位置を表すものです。もうひとつはローカル座標で，これは各グラフごとの座標系です。またまた，嘘を承知でグラフ＝ウィンドウと考えてもらうと，ローカル座標は，ウィンドウの（タイトルバーを除いた）左上隅が(0,0)となるような座標系です。グラフに対する描画時の座標指定にはローカル座標を使います。ローカル座標を使うかぎり，ウィンドウが実画面上のどこに表示されていようと，気にする必要はないわけです。

座標は上位ワードをx座標，下位ワードをy座標とする32ビットのデータで表現されます。ただし，グラフマネージャにおいてこの座標が意味するのはドットではなくポイントです。ポイントは大きさを持たない真の意味での点であり，各ドットの左上隅を指しています。対して，ドットは縦横1単位の面積を持ちます。

どうしてこうなっているかというと，MacintoshのQuickDrawがそうだったからです。では，QuickDrawがなぜこんな仕様になっているのかというと，画面上での図形の長さを数学的に表したかったからのようです。水平な線分の長さは，数学では両端点のx座標の差で求められます。ところが，X-BASICのline関数を考えてみると線分の長さは両端点のx座標の差よりも1ドット分長くなりますね。QuickDrawの開発者はこれを嫌い，座標の差がそのままドット数に反映されるようにしたのでした。

●描けるもの

グラフマネージャは，以下のようなさまざまな図形を描くことができます。

- 直線
- 長方形（rectangle，レクタングル）
- 角丸長方形（rounded rectangle，角の丸まった長方形）
- 真円を含む楕円（oval）
- 自由な形の領域（regeon，リージョン）
- パターン（イメージ）
- 文字，文字列

レクタングルなどの場合は輪郭（frame，フレーム）のみを描くか，塗り潰す（fill，フィル）かを選べるのはもちろん，いわゆるペンパターン，フィルパターン，描画モードを指定することもできます。

なお，MacintoshのQuickDrawでは，このほかに円弧と多角形，さらにシードフィルによるペイントがサポートされていますが，SX-WINDOWにはありません。一応“予約”になっていますから，そのうちサポートされるのでしょうか。

■レクタングル

繰り返します。レクタングルとは長方形のことです。「それなら長方形と書け！」と怒られそうですが，“××レクタングル”のような複合語を“××長方形”と表記するのでは間が抜けているので，以下，レクタングルで押し通します。

レクタングルは，左上隅の座標と右下隅の座標を表す2つのポイントを並べた形で表現されます。レクタングルの縦横の長さは座標の差でそのまま表されるのは前述のとおりです。左上と右下のx，y座標のどちらか少なくとも一方が等しい場合には，レクタングルは空になります（これをヌルレクタングルと呼びます）。

また，グラフマネージャでは，レクタングルの座標は必ず，左上，右下の順序に並べる約束になっています。そうなっていない場合は，ヌルレクタングル扱いです。

■リージョン

リージョンは単純な図形では表せないような複雑な形を表現するものです。輪郭が閉じてさえいれば，リージョンの形には制

限がありません。ドーナツ状に穴が空いていても，2つ以上の部分に分割されていてもかまわないのです。

リージョンは通常，レクタングルや楕円の描画ファンクションを利用して作成します。つまり"以下の図画はリージョンの定義だ"と宣言してから，ラインや図形のフレームを描くSXコールを呼び出すと，図形が描かれる代わりにリージョンの内部データが作成されていくのです。また，2つのリージョン間で差や和，積び，変わりをとることで，単純なリージョンから複雑なリージョンを作り出すこともできます。ただし，リージョンは必ず閉じていなければならないことになっていますので，特にラインを使ってリージョンを作成するときには，始点と終点を忘れずに結んでおく必要があります。

なお，リージョンはリロケータブルブロックに置かれ，ハンドルによって参照されます。

●グラフスクリプト

グラフマネージャはグラフスクリプトと呼ばれるデータを扱うこともできます。これはMacintoshでいうpictureに相当し，複数の図形の描画を行う描画コマンド列です。つまり，図形をビットイメージで保持するのではなく，直線やレクタングルなど（を描くコマンド）の組み合わせで表現するものです。

グラフスクリプトの各描画コマンドは内部コードで表されますが，凝ったことをしない限り，内部コードを知る必要はありません。リージョンの作成同様，描画を行い，その描画手順を記録することでグラフスクリプトを作成することができます。

グラフスクリプトもまたリロケータブルブロックに置かれ，ハンドルによって参照されます。

図6　ウィンドウ構造体

```
        offset 0
pix             ds.b    graph   * グラフマン用[illegible]ストラクチャ
wKind           ds.w    1       * ウィンドウの種類
wVisible        ds.b    1       * 表示状態
wHilite         ds.b    1       * アクティブ状態
wClose          ds.b    1       * クローズボックス
wStatus         ds.b    1       * ウィンドウマネージャ内部で使用
wOption         ds.w    1       * ウィンドウアイテムの指定
wOutside        ds.l    1       * ウィンドウの外枠を示す region
wInside         ds.l    1       * ウィンドウの内部を示す region
wUpdate         ds.l    1       * ウィンドウの書き直しを示す region
wDef            ds.l    1       * ウィンドウ定義関数への handle
wDefData        ds.l    1       * ウィンドウ定義関数のデータ
wTitle          ds.l    1       * ウィンドウタイトルへの handle
wTWidth         ds.w    1       * ウィンドウタイトルの幅
wControl        ds.l    1       * コントロールへの handle
wNext           ds.l    1       * 次のウィンドウへの pointer
wPicture        ds.l    1       * 自動アップデート用 graph script
wTask           ds.l    1       * SXシェル用タスクID
window
```

●文字の描画

グラフマネージャはX68000がROMに持っているすべてのフォントを扱うことができます。また，ユーザー定義フォントを使う方法も用意されているようです。

文字描画時には，

- 強調
- 斜体
- 下線
- 袋文字（中抜き）
- 影つき袋文字

といった文字飾りを利用することができます。文字飾りは当然，組み合わせて指定できます。反転がありませんが，これは描画モードで，または，描画色と背景色を逆にすることで実現できます。

文字は128×128ドット以内であれば，任意の大きさに縮小・拡大して表示することもできます。ただし，このあたりにはちらほらとバグが残っています。

ウィンドウ

■ウィンドウの各部名称

ウィンドウは大きく分けて，アウトサイドリージョンとインサイドリージョンの2つの部分から構成されます。アウトサイドリージョンはタイトルバーや，輪郭といったウィンドウの外枠部分です。インサイドリージョンは，それ以外の，実際に文字や図形を表示するウィンドウの中身の部分となります。

標準ウィンドウのタイトルバーには，右端にウィンドウを閉じるためのボタン，クローズボックスがあります。そのほか，ディレクトリを表示するウィンドウでは，タイトルバー上にクリップ，ドライブ名を表示する枠，親ディレクトリに戻る矢印型のボタンなどの，オプションアイテムが並びます。

インサイドリージョンには，ときにスクロールバーや，ウィンドウのサイズを変更するグローボックスが配置されます。とりあえず，これらが特別な領域ではなく，インサイドリージョン中に"置かれているだけ"ということを頭に入れておいてください。なお，細かな話になりますが，スクロールバーはコントロールに，グローボックスはクリップなどと同様，ウィンドウのオプションアイテムに分類されます。前者はコントロールマネージャの，後者はウィンドウマネージャの，それぞれ管理下にあるわけです。

●ウィンドウ構造体

各ウィンドウはそれぞれひとつのウィンドウ構造体を持ち，ウィンドウマネージャはこのウィンドウ構造体によりウィンドウを管理します。ウィンドウ構造体の具体的な構造を図6に示しておきます（部品マニュアルからの抜粋です）。ウィンドウ構造体はユーザープログラムから直接書き換えるべきものではありませんから，各フィールドの意味を詳しく知っている必要はありませんが，少しだけつまみ食いしてみましょうか。

頭のpixフィールドはウィンドウの描画領域であるグラフ構造体そのものです。ここにグラフ構造体が埋め込まれていることにより，グラフ構造体に対する操作（つまりは描画）をウィンドウ構造体にそのまま適用することができるようになっています。

少し飛んで，wUpdateにはウィンドウ中，再描画しなければならないリージョンへのハンドルが格納されます。ウィンドウを描き直すのはアプリケーション側の仕事です。wUpdateはウィンドウの前後関係が変わったときなどにウィンドウマネージャによって更新されます。ユーザープログラムはアップデートイベント（画面を描き直す必要があるという合図）を受け取ったとき，wUpdateで指定されるリージョン内を再描画することになります。

また飛んで，wControlにはウィンドウ上に配置されたコントロールへのハンドルが収められます。こうやって，ウィンドウとコントロールを結びつけるわけです。

その下のwPictureにはグラフスクリプトをぶら下げることができます。このフィールドにグラフスクリプトへのハンドルを格納しておくと，アップデートイベント発生時に，自動的にそのグラフスクリプトを使ってウィンドウの描き直しが行われます。その場合，ウィンドウの持ち主であるタスクには，アップデートイベントは報告されません。この機能は，決まりきった内容しか表示しないウィンドウでは有効な手抜き技となるでしょう。しかし，です。残念ながらSX-WINDOW Ver.1.00～1.02ではこの機能は死んでいます（一度でも試してみれば，発見できるバグだと思うのですが）。

■ウィンドウ定義関数と定義ID

SX-WINDOWを使っていると，何種類ものウィンドウがあるのがわかります。ふつうのウィンドウに混じって，タイトルバーしかないような時計Xのウィンドウとか，電卓Xのウィンドウ，また，ダイアログにも何通りかがありますね。

ウィンドウの種類はウィンドウオープン時にウィンドウ定義IDで指定します。ウィンドウ定義IDは，ウィンドウ定義関数と密接な関係があります。先ほども簡単に触れましたが，ウィンドウ定義関数はウィンドウの制御や枠の描画を行うモジュールです。ウィンドウの種類ごとに，リソースタイプWDEFを持つリソースとして用意されています。表2に現在SX-WINDOWで使えるWDEFの一覧を示しておきましょう。

さて，ウィンドウ定義IDは，WDEFのリソースID×16＋4ビットのオプションコードの16ビットデータです。標準ウィンドウの場合，オプションコードは1ビットごとに意味を持ち，グローボックスなどのオプションアイテムの有無やON/OFFを指定します。本当はオプションコードは16ビットなのですが，ウィンドウ定義IDでは，その下位4ビットのみを指定します。残りはオープン後，ユーザーアプリケーション側でウィンドウ構造体のwOptionフィールドに設定することになっています（wOptionを設定するSXコールはありません）。なんか奇妙な仕様です。

SXシェル上のプログラム

■タスク

SXシェル上（正確にはタスクマネージャ上というべきでしょうか）では，各ウィンドウ1個1個がタスクとなります。ディレクトリ内容を表示しているウィンドウも，デスクアクセサリ類もそれぞれがひとつの完結したタスクです。とはいっても，ピンボールのように複数のウィンドウを開くタスクもあれば，逆に，ウィンドウを持たないタスクも考えられますから，ウィンドウ＝タスクではありません（なんか変な説明）。そして，SXシェルもまた，タスクのひとつです。

各タスクにはそれぞれタスク管理テーブルが用意されており，タスクマネージャはこれによりタスクを管理します。各タスクはタスクIDと呼ばれる値で識別されます。タスクIDはタスク管理テーブル，そしてウィンドウ構造体中のwTaskフィールドに収められ，タスクとウィンドウを結びつけています。特にタスクIDが0であるタスクはスーパーユーザーと呼ばれ，タスクマネージャを初期化したりといった雑用を負わされます。通常はSXシェルがスーパーユーザーとなりますが，COMMAND.Xから直接起動しても動くデスクアクセサリ，アプリケーションは，ときにスーパーユーザーとなりうるわけです。

タスクはタスクマネージャのSXコールTSForkなどによって生成され，TSExitにより消滅します。なお，"タスクの生成"と"プログラムの起動"は，概念上はまったく別物ですが，少なくともSX-WINDOWシステムでは，ほぼ同時に起こります。"タスクの消滅"と"プログラムの実行終了"も同様です。

●モジュールヘッダ

SXシェルの上でタスクとして動作するプログラムは，かならずモジュールヘッダと呼ばれるデータブロックを持たなければなりません。その構造は図7のとおりです。モジュールヘッダはRファイルではファイルの先頭，Xファイルではendで指定したプログラムの実行開始アドレスの直前に置く約束になっています。

モジュールヘッダの先頭，モジュールタイプは，プログラムの性質（?）を表します。モジュールタイプには，

・リエントラントなプログラム
・リエントラントではないプログラム
・同時に複数実行できないプログラム

の3種類があり，ソース中ではSXCALL.MAC中で定義されたObjectR，ObjectC，ObjectOという記号定数でそれぞれのタイプを表現します。リエントラントとはなにかは，ひとまず横に置いておいて，先に進みましょう。

モジュールサイズは，プログラムのコードサイズです（定数データなども含みます）。この値はR形式のファイルでのみ意味を持ちます。

次のスタートアドレスオフセットは，モジュールヘッダの先頭アドレスから"TSForkによりプログラムが起動されるときの実行開始アドレス"までのバイト数です。

そして，データサイズが続きます。プログラム起動時には，ここに入れておいたバイト数分の作業用メモリがノンリロケータブルブロックとしてヒープ上に確保され，プログラムには，そこへのポインタがa1レジスタで渡されます。システムがワークエリアを自動的に確保してくれるわけです。

■リエントラントなプログラム

リエントラント（re-entrant，再入可能）というのは，"ルーチンが複数の場所から非同期的に呼び出されても正常に動く"ことを表現する言葉です。サブルーチンの場合でいうと，"サブルーチンからリターンする前に，同じサブルーチンがどこかからさらに呼び出されても大丈夫"であることが保証されていれば，リエントラントといいます。たとえば，再帰する関数（サブルーチン）は，リターン前に自分で自分のことを呼び出す形になりますから，リエントラントであることが要求されます。

表2　ウィンドウ定義関数(WDEF)

ID 16	時計 X用
ID 32	標準ウィンドウ
ID 36	ダイアログボックス（影つき）
ID 38	ダイアログボックス（プレーン）
ID 38	ダイアログボックス
ID 40	ダイアログボックス
ID 48	標準ウィンドウ（グラフィック使用可）
ID 50	標準ウィンドウ（グラフィック使用可　クローズボックス［illegible］）
ID 64	電卓 X用

図7　モジュールヘッダ

```
        .offset 0
mdType          ds.l            * モジュールタイプ
mdSize          ds.l            * モジュールサイズ
mdStOffSet      ds.l    1       * スタートアドレスオフセット
mdDataSize      ds.l    1       * データサイズ
mdRsv           ds.l    4       * システム予約
mdEntry                         * Xファイル時のエントリー（ end で指定する）
mdHead
```

プログラムをリエントラントにするためには、最低限、複数の呼び出し側同士がワークを踏み換えあって競合する事態を避けなければなりません。ワークエリアはローカルなものでなければならないのです。具体的には、絶対アドレスでメモリを踏み換えることを避け、ワークはスタックかヒープ上に確保します（定数データは絶対アドレスで参照してもかまいません）。

さて、SXシェル上のプログラムに目を向けましょう。ディレクトリ表示ウィンドウを例に挙げるまでもなく、ときには同じプログラムを複数動作させる場合・させたい場合があります。このとき、プログラムをタスクの数だけメモリ上に置くのは明らかにヒープの無駄遣いです。ワークエリアだけ、タスクごとに用意してやれば、プログラムの実行コード自体は共有できそうに見えませんか。プログラムがリエントラントに作られていれば、それが可能になるのです。

モジュールタイプがOBJECT.Rのプログラムを起動するとき、タスクマネージャはすでに同じプログラムがメモリ上に存在するかどうかを調べます。もし、存在しなければプログラムを読み込み、同時にモジュールヘッダで示されたデータエリアを確保します。存在するのであれば、データエリアだけを確保して、プログラム本体は読み込みません。プログラム本体の分だけメモリが節約できることになります。同時に複数動かすプログラムは、なるべくリエントラントに作りたいものです。

なお、タスクマネージャはモジュールタイプにのみ頼ってプログラムがリエントラントかどうかを判断します。実際にはリエントラントでないプログラムにObject.Rのモジュールタイプを与えた場合の動作は保証されません。もっとも、これを逆手にとれば、ワークの一部を複数のプログラムから共用することもできます。

■タスクマネージャからのイベント取得

アプリケーションプログラムはメインループの中で、必ず1度タスクマネージャにイベントを要求します。イベントを要求するには、イベントに関する情報を格納するイベントレコードへのポインタと、イベントマスクをスタックに積んで、SXコールTSEventAvailを呼び出します（このときタスクの切り替えが行われますが、アプリケーションは気づきません）。イベントマスクは、自分にとって不要なイベントに煩わされないようマスクするものです。TSEventAvailからは、マスクしなかったイベントのうち、最も優先順位の高いイベントが、イベントコードと付随情報の形でイベントレコードに返されます。

タスクマネージャから得られるイベントには以下のものがあります。

- ヌルイベント
- マウス左ボタンダウンイベント
- マウス左ボタンアップイベント
- マウス右ボタンダウンイベント
- マウス右ボタンアップイベント
- キーダウンイベント
- キーアップイベント
- アップデートイベント
- アクティベートイベント
- システムイベント1
- システムイベント2
- ユーザーイベント1
- ユーザーイベント2

アクティベートイベントまではイベントマネージャの管轄です。タスクマネージャは、イベントマネージャから送られたイベントもまとめてアプリケーションに返します。

ヌルイベントは、何もイベントがないときに定期的に発行されるイベントです。擬似マルチタスクの鍵でしたね。

マウスボタンとキーのアップ・ダウンイベントについては触れるまでもないでしょう。それぞれ、ボタン・キーが押されたとき、離されたときに発生するイベントです。

アップデートイベントはさっきも顔を出しました。ウィンドウの前後関係が変化したなどの理由により、いままで見えなかった部分が見えるようになったから、ウィンドウをアップデート（再描画）しなさいというイベントです。ウィンドウの枠（アウトサイドリージョン）の再描画はウィンドウマネージャ（というかウィンドウ定義関数）が行いますから、ユーザープログラム側では、ウィンドウの中身（インサイドリージョン）のみを描き直すことになります。なお、アップデートイベントは非常に優先順位の低いイベントです。ほかの重要なイベントがない場合にのみ送られてきます。これには、画面の再描画をなるべくまとめて行うことで、全体の処理効率を上げる意味があります。

アクティベートイベントは、ウィンドウのアクティブ状態が変化したときに発生するイベントです。どのウィンドウがアクティブになったかという情報が同時に与えられますので、それが自分のウィンドウであれば自分をアクティブ状態にし（それを覚えておく）、そうでなければ非アクティブ状態にすることになります。アクティベートイベントは優先順位の高いイベントで、ウィンドウの状態が変化したときには、マウスボタンの押し下げやキー入力がイベントキューに残っていたとしても、即座に発行されます。

システムイベント1、2はタスクマネージャからのイベントです。具体的な内容はイベントレコードにタスクマネージャのイベントコードとして返されます。システム1は全タスクに対するイベント、システム2は各タスクに個別に送られるイベントということになっていますが、中身は同じです。タスクマネージャからのイベントには、タスクを終了せよという指示など、各プログラムが必ずサポートしなければならない重要なものが含まれます。

また、タスクマネージャのイベントコード中128以降はアプリケーションに解放されています。これらのコードに適当な意味を持たせ、アプリケーション側からイベントを発行することでタスク間通信が行えます。

ユーザーイベントはユーザー用です。いまのところは使われていません（きっと）。

サンプルプログラム

まだまだ説明が不十分な気もしますが、そろそろプログラミングに進み、細かな部分はサンプルプログラムを見ながら解説したいと思います。

リスト1はウィンドウを開き、文字列を適当に表示するプログラムです。実行ファイルは、

```
A>AS WINTEST
A>LK WINTEST
```

により作成します。付録ディスク中のインクルードファイルをがりがり取り込んでいますから、AS Ver.1.0xを使っている人はカレントディレクトリに、Ver 2.0を使っている人は環境変数includeで指定したディレクトリに、各インクルードファイルを置いておいてください。

では、プログラム各部の意味、動作を説明します。

■マクロ定義（32～41行）

2つのマクロを定義しています。ひとつ目のSXCALLはSXコールを呼び出すだけの簡単なマクロです。べつに、dc.wでコール番号を埋め込んでも、DOSCALL.MACで定義されているマクロDOSを使ってもよかったのですが、ほんの気分であります。

2つ目のLASCIIは、簡易マニュアルでLASCII文字列と呼ばれている形式の文字列データを定義するマクロです。LASCIIは文字列の頭に1バイトで文字列長を置く形式で、LASCII形式の文字列をdc.bで用意するときには、自分で文字列の長さを数えて先頭に置く必要があります。それが嫌だったので、マクロにして、文字列の長さはアセンブラに計算させるようにしてみました。

■データエリアの定義（45～63行）

データエリアの大きさと、その内部構造をoffset擬似命令で定義しています。起動時に確保され、a1レジスタで渡されるデータエリアに対するアクセスは、ここで定義したオフセットを使い、ディスプレイスメントつきアドレスレジスタ間接形式で行います。

WINTEST.Xでは、本当に最小限のワークしか用意していません。

wmptrにはウィンドウをオープンしたときに返されるウィンドウ構造体へのポインタを格納します。以降、ウィンドウを指定する必要があるときには、このポインタを使います。また、リストではwinptrをウィンドウがオープンされているかどうかのフラグに兼用しています。winptrが0ならウィンドウは未オープンです。

winsizeはウィンドウオープン時に、ウィンドウの位置と大きさを指定するのに使うレクタングル構造体です。

eventrecはイベント取得時に使うイベントレコードで、その先頭アドレスをTSEventAvailに渡すことになります。

activeflagは自分が現在アクティブウィンドウかどうか（最前面に表示されているかどうか）を覚えておく1バイトのフラグです。非0ならアクティブと決めます。

それ以降はスタック領域です。すっかり説明し忘れていましたが、データエリアが確保された際に、sp（usp）はデータエリア末尾を指すようタスクマネージャによって初期化されます。データエリアにはその分も用意しておかなければなりません。

■モジュールヘッダ（68～65行）

このプログラムはリエントラントです。以下、コードサイズ、スタートアドレスオフセットはラベルの差で、データエリアの大きさはoffsetで定義した記号定数で指定しています。

■exec時の処理（71～133行）

COMMAND.Xから直接起動した場合には、71行から実行が始まります。このアドレスは52行のend擬似命令で指定してあります。多少凝ってあり、アセンブル時に

```
A>AS /sNDEBUG WINTEST
```

のようにシンボルNDEBUGを定義すると75～81行の部分がアセンブルされ、さもなくば85～131行がアセンブルされるようになっています。NDEBUGを定義した場合には、COMMAND.Xからは起動できない旨のメッセージを出してWINTEST.Xは終了します。NDEBUGが定義されていない場合は、デバッグモードということで、デバッグ用のシェルSXWDB.Xを起動し、WINTEST.Xはその上で走ります（SXWDB.Xは開発キット中に含まれます）。SXWDB.Xはアプリケーションに代わって、スーパーユーザーの役割を果たしてくれます。

85行以下、SXWDB.Xを起動するところまでは、子プロセスの生成の常套手段です。スタックポインタを初期化してから、DOSコールsetblockで不要なメモリを切り離し、DOSコールexecでSXWDB.Xを起動します。SXWDB.Xはpathの通ったディレクトリに置いておくようにしてください。

SXWDB.Xは起動後、WINTEST.XをTSFockを使って再起動します。そして、制御はWINTEST.Xに移り、142行から再スタートします。

■TSFock時の処理（142～154行）

TSFockにより起動されたプログラムにはレジスタを介していくつかの情報が渡されます。リスト1で使っているのは、

a1　データエリアへのポインタ
a2　コマンドライン引数
d0　自分のタスクID

です。

a1は以降のデータエリアアクセス時のベースアドレスとなりますから、適当なアドレスレジスタ（リスト1ではa5）に転送しておきます。

続いて、アクティブかどうかのフラグと、ウィンドウ構造体へのポインタを0で初期化しています。データエリアの初期値は不定ですから、必要であれば、自分で初期化しなければなりません。

その後、d0が負かどうかを調べています。d0が負の場合は、起動途中でエラーが発生したことを意味しています。WINTEST.Xはエラー時にはそのまま終了処理に移ります。

■初期化（472～529行）

サブルーチンinitでは、ウィンドウの表示を行います。

まず、SXコールTSTakeParamを使って、a2レジスタで渡されたコマンドライン引数からウィンドウ表示座標を取り出します。SXシェル上で"画面状態を保存"したとき保存されたウィンドウの位置が、

-w nnn,nnn,nnn,nnn

の形式で与えられる場合があるからです（SYSDTOP.SXをダンプして納得してください）。この指定があった場合、TSTakeParamはd0の最下位ビットを1にして戻ります。このビットが1であれば、winsizeにはウィンドウの位置が格納されていますから、ウィンドウのオープン処理に移りま

リスト1

```
 1 *
 2 *        ウィンドウを開いて文字を表示する
 3 *
 4         .include        doscall.mac
 5         .include        sxcall.mac
 6         .include        sxcall.equ
 7 *
 8 TRUE            equ     -1                      *真
 9 FALSE           equ     0                       *偽
10 *
11 CR              equ     $0d                     *復帰コード
12 LF              equ     $0a                     *改行コード
13 *
14 WDEFID          equ     WF_STD                  *ウィンドウ定義関数ID
15                                                 *（標準ウィンドウ）
16 WINOPT          equ     WC_GBOX|WC_GROOM
17                                                 *ウィンドウオプション
18                                                 *（グローボックスあり＆ON）
19 WINOPTLOW       equ     WINOPT&$ffff            *ウィンドウオプション下位4ビット
20 WDEFID2         equ     WDEFID<<4+WINOPTLOW
21                                                 *ウィンドウ定義ID
22 *
23 WINX            equ     200                     *ウィンドウ横ドット数
24 WINY            equ     160                     *ウィンドウ縦ドット数
25 WINSIZE         equ     WINX<<16|WINY           *ウィンドウの大きさ
26 *
27 EVENTMASK       equ     EM_MSLDOWN|EM_UPDATE|EM_ACTIVATE|EM_SYSTEM1|EM_SY
28                                                 *イベントマスク
29 *
30 C_PLATE         equ     0                       *プレーンの番号
31 *
32 SXCALL  .macro  callno                          *SXコールマクロ（ほんの気分）
33         dc.w    callno                          *
```

す。

位置の指定がなかった場合は、TSGetWindowPosを使って、タスクマネージャに適当な位置を決めてもらいます。TSGetWindowPosはd0に適当な座標（上位ワードにX座標、下位ワードにY座標）を返しますから、この座標をウィンドウの左上隅の座標とし、ウィンドウの大きさを加えて右下隅の座標を求め、それぞれwinsizeに格納します。そして、492行に合流して、ウィンドウをオープンする態勢に入ります。

492行では、まず自分のタスクIDを得ています。これはウィンドウをオープンするときに必要な情報です。起動時のd0にもタスクIDは格納されていたわけですが、ごちゃごちゃやっている間にd0は破壊されていますから、TSGetIDで取得しなおしています。

以下、ウィンドウをオープンするSXコールWMOpenに与える引数をどんどんスタックに積んでいきます。順に、
- タスクID
- クローズボックスの有無（非0ならあり）
- ウィンドウをどのウィンドウの後ろに置くかの指定（−1を指定すると一番手前）
- ウィンドウ定義ID（標準ウィンドウで、グローボックスあり）
- ウィンドウを表示するかどうか（非0なら表示）
- ウィンドウのタイトル（LASCIIへのポインタ）
- ウィンドウの位置（レクタングル構造体へのポインタ）
- ウィンドウ構造体を格納するメモリアドレス（0ならメモリマネージャに確保してもらう）

です。ウィンドウ定義IDは14〜20行のあたりで記号定数に定義してあります。14行のシンボルWDEFIDの値を変えればいろいろなタイプのウィンドウを表示することができますし、WINOPTの値を変更してもそれなりに遊べます。気が向いたら試してみてください。

WMOpenから戻ってきた時点でNフラグが立っているときはエラーです。正常にオープンできたときにはa0にウィンドウ構造体へのポインタが格納されていますから、ワークにしまっておきます。

それから、510行でウィンドウオプションをウィンドウ構造体に直接書き込んでいます。WMOpenを呼んだときに"グローボックスあり"までは指定できているのですが、グローボックスはONに設定しないと

```
        endm
*
LASCII  macro   string
        local   top,bot
        dc.b    bot-top
top     dc.b    string
bot
        endm
*
*       データエリアの定義
*
        offset  0
*
wingptr:        ds.l    1       *ウィンドウ構造体へのポインタ
winsize:        ds.b    rect    *ウィンドウの位置
evrecord:       ds.b    Event   *タスクマネージャ用のイベントレコード
activeflag      ds.w    1       *自分がアクティブかどうかのフラグ
                even
                ds.l    256     *スタック
sizeof_WORK                     *データエリアの総バイト数
*
        text
        even
*
*       モジュールヘッダ
*
head
        dc.l    OBJECTR         *リエントラント
        dc.l    Exit-head       *実行コードのバイト数
        dc.l    start-head      *スタートアドレスオフセット
        dc.l    sizeof_WORK     *データエリアのバイト数
        dc.l    0,0,0,0         *システム予約
*
*       EXEC時のエントリ
*       COMMAND Xから起動したときはここから始まる
*       (そのように、endで指定しておく)
*
exec_entry:

  ifdef [illegible]

        pea.l   errormes(pc)
        DOS     _PRINT
        DOS     _EXIT

errormes
        dc.b    'SXシェル上で起動してください' CR LF 0
        even

  else

        lea.l   inisp(pc),sp            *sp初期化

        lea.l   16(a0),a0               *余計なメモリを切り離す
        sub.l   a0,a1                   *
        move.l  a1,-(sp)                *
        move.l  a0,-(sp)                *
        DOS     _SETBLOCK               *

        clr.l   -(sp)                   *SXカーネルをpathから探して
        pea.l   cmdarg(pc)              *  起動する
        pea.l   fname(pc)               * (このプログラム自体は
        move.w  #2,-(sp)                *  SXカーネルから再起動される)
        DOS     _EXEC                   *検索
        tst.l   d0                      *SXカーネルを見つけた?
        bmi     error                   *  見つからなかった
        clr.w   (sp)                    *
        DOS     _EXEC                   *起動
        tst.l   d0                      *SXカーネルを起動できた?
        bmi     error                   *  起動できなかった

        DOS     _EXIT                   *正常終了
*
error   pea.l   errmes(pc)              *SXカーネルが起動できなかった
        DOS     _PRINT                  *
        move.w  #1,-(sp)                *エラー終了
        DOS     _EXIT2                  *

*
*       EXEC時用データ／データ
*       SXシェル上から起動したときはこれらは捨てる
*
        data
        even
*       [illegible]
fname   dc.b    [illegible]             *SXカーネルを起動するコマンド
                                        *(とりあえずデバッグ用の[illegible])
fname2  ds.b    [illegible]             *(フルパスで上書きされる[illegible])
                                        *  [illegible]バイト必要)
errmes  dc.b    'SXカーネルが起動できませんでした' CR,LF,0
                                        *エラーメッセージ
*
        bss
        even
*
cmdarg  ds.b    256                     *コマンドライン作業用
        ds.b    [illegible]             *初期スタック
inisp

        end
```


見慣れたあの形では表示されないのです（510行を削って確かめてみましょう）。

そして、GMSetGraphで自分のウィンドウ（に対応したグラフ）をカレントグラフにしてから、WMDrawGBoxでグローボックスを描きます。グローボックスはインサイドリージョンに描かれますから、表示・再表示はアプリケーション側の仕事です。

これでウィンドウの表示が終わりました。描かれるのは"スクロールバーなしでグローボックスだけがある"という多少邪道なウィンドウです。なお、ウィンドウの中身は、あとでアップデートイベントが発生したときに描きます。

■メインループ（158〜176行）

ずーっと戻って、メインループです。イベントレコードとイベントマスクをTSEventAvailに渡して、イベントを得ます。イベントマスクは27行で記号定数EVENTMASKに定義してあります。
- マウス左ボタンダウン
- アップデート
- アクティベート
- システム1、2

という必要最小限のイベントのみを受け取るようにしています。

TSEventAvailから戻った時点で戻り値のd0がハードウェアエラーによるアボートを示していた場合は終了します。

エラーがなければ、イベントレコードのmWhatフィールドにイベントコードが格納されています。そこで、ジャンプテーブルにより、イベントに対応した処理ルーチンをサブルーチンコールします。

処理がすんで戻ってきたら、また、ループの頭に飛んで、次のイベントを要求します。これがえんえんと続くのです。

●マウス左ボタンダウンイベント発生時の処理（208〜244行）

208〜244行がマウスの左ボタンが押された場合の処理です。イベントレコードのmWhereフィールドには、どのウィンドウでマウスボタンが押されたかが、ウィンドウ構造体へのポインタの形で格納されていますから、まず、自分のウィンドウかどうかを調べます。違えば関係ありませんから即戻ります。

自分のウィンドウ上だということがわかったら、現在、自分がアクティブウィンドウかどうか、activeflagを調べてみます。アクティブウィンドウかどうかで、処理が多少違うのです。ソースとは順序が逆ですが、先にアクティブウィンドウだった場合の処理を見てもらいましょう。231行からです。

```
*
        text
*
*       TSEXEC時のエントリ
*
WORK    reg     a6                      *データエリアのベースアドレス
*
start:
        move.l  a1,WORK                 *WORK=a6
                                        *以下データエリアはWORKからの
                                        *  オフセットでアクセスする
        sf.b    activeflag(WORK)        *アクティブフラグをクリア
        clr.l   wingptr(WORK)           *ウィンドウ構造体へのポインタを
                                        *  クリア(ウィンドウ未オープン)

        tst.l   d0                      *d0が負なら
        bmi     exit                    *  起動途中でエラーが発生した

        bne     init                    *初期化する
        bmi     exit                    *-1なら初期化に失敗した
*
*       メインループ
*
mainloop
                                        *イベントを得る
        pea.l   evrecord(WORK)          *イベント格納用レコード
        move.w  #EVENTMASK,-(sp)        *イベントマスク
        SXCALL  __TSEventAvail          *イベントを得る
        addq.l  #6,sp                   *(イベントキューからは取り除かない)
        cmpi.w  #HE_ABORT,d0            *エラー?
        beq     exit                    *  そうなら終了
                                        *イベントに応じた処理ルーチンに分岐する
        move.w  evrecord+EVR_mWhat(WORK),d0
                                        *d0=イベントコード
        andi.w  #$000f,d0               *念のためマスク
        add.w   d0,d0                   *分岐先への
        move.w  jmptbl(pc,d0.w),d0      *  相対距離を求める
        jsr     jmptbl(pc,d0.w)         *対応ルーチンを呼び出す

        bra     mainloop                *えんえんと繰り返す

*
*       イベントに対応した処理ルーチンへのジャンプテーブル
*
jmptbl
@ = jmptbl
        dc.w    noop-@                  *ヌル
        dc.w    mouseldown_event-@      *マウス左ボタンダウン
        dc.w    noop-@                  *マウス左ボタンアップ
        dc.w    noop-@                  *マウス右ボタンダウン
        dc.w    noop-@                  *マウス右ボタンアップ
        dc.w    noop-@                  *キーダウン
        dc.w    noop-@                  *キーアップ
        dc.w    update_event-@          *アップデート
        dc.w    noop-@                  *       (システム予約)
        dc.w    activate_event-@        *アクティベート
        dc.w    noop-@                  *       (システム予約)
        dc.w    noop-@                  *       (システム予約)
        dc.w    system1_event-@         *システム1
        dc.w    system2_event-@         *システム2
        dc.w    noop-@                  *ユーザー1
        dc.w    noop-@                  *ユーザー2

*
*       自分に無関係なイベントの処理
*       (マスクしているから、本当はここにはこないが念のため)
noop
        rts

*
*       マウス左ボタンダウンイベントの処理
*
mouseldown_event
                                        *自分のウィンドウの上かどうか調べる
        move.l  wingptr(WORK),a1        *ウィンドウ構造体へのポインタ
        cmp.l   evrecord+EVR_mWhere(WORK),a1
                                        *自分かな?
        bne     LDrstn                  *違った
                                        *自分のウィンドウの上だった
        tst.b   activeflag(WORK)        *もうアクティブだった?
        bne     callWM                  *前からアクティブだった
                                        *アクティブにする
        move.l  a1,-(sp)                *自分のウィンドウを
        SXCALL  __WMSelect              *  一番上にする
        addq.l  #4,sp                   *
                                        *タイトルバーの上かどうか調べる
        move.l  evrecord+EVR_mWhere(WORK),-(sp)
                                        *
        SXCALL  __WMFind                *どこでボタンが押されたか?
        addq.l  #8,sp                   *
        cmpi.w  #M_INDRAG,d0            *タイトルバーの上?
        bne     LDsblp                  *  違った
        SXCALL  __SMLStill              *まだボタンは押されている?
        beq     LDsblp                  *  もう離された

callWM: pea.l   evrecord(WORK)          *あとはウィンドウマネージャに
        move.l  a1,-(sp)                *  まかせてしまう
        SXCALL  @SCallWindM             *
```

本来，ここでやらなければならないことはたくさんあります。まず，WMFindを使って，ウィンドウのどの位置でマウスボタンが押されたか調べます。結果はパートコードというヤツで返されますから，それに応じて処理を振り分けます。

タイトルバーだったらマウスボタンが離されるまでのあいだマウスの動きを追跡しつつ，点線で描いたウィンドウの枠を動かし，ボタンが離されたその位置にウィンドウを移動します。

クローズボックス上であれば，マウスカーソルがクローズボタン上にあるあいだ，クローズボックスをハイライト表示し，ボタンが離されたときに，まだマウスカーソルがクローズボックス内であれば，ウィンドウを閉じます。そうでなければ，何もしてはいけません。ユーザーは途中で気が変わったのです。

グローボックスがドラッグされた場合はやはりマウスカーソルを追跡し，ボタンが離されたら，ウィンドウの大きさを変更します。また，グローボックスがダブルクリックされた場合はズームですから，ウィンドウを最大に拡大（または元に戻す）します。

とても面倒ですね。Macintoshでは実際にこういうことをやっているようです。SX-WINDOWにも該当コールがありますから，同じことをやろうと思えばできます。でも，タスクマネージャに任かせてしまうという技があるのでした。

リスト1の233行で呼び出しているSXCallWMが上に挙げた一切の処理を受け持ちます。イベントレコード（その中にボタンが押された座標を収めたフィールドがあります）と，ウィンドウ構造体へのポインタを渡すだけという簡便さです。最終的にSXCallWMは，マウスのボタンが離された位置のパートコードを返します。それがクローズボックスを表していたら，終了処理に飛びます。

では，非アクティブだった場合に戻りましょう。224行からです。まず，WMSelectにより，自分を最前面に出し，アクティブウィンドウにします。それから，WMFindにより，ボタンが押された位置のパートコードを得て，それがタイトルバーを意味していた場合にのみ，SXCallWMを呼び出し，ウィンドウのドラッグを任せます。下敷になっていたウィンドウをいきなりドラッグすることを許しているわけです。

マウス左ボタンダウンイベント処理の最後では，TSGetEventにより，いま処理したマウス左ダウンイベントをイベントキューから取り除きます。TSEventAvailはイベントキューを覗き見るだけで，実際にはイベントを取り除きません。関係のないイベントまで取り出してしまっては，ほかのプログラムが困りますから，自分で処理した場合にのみ，TSGetEventを呼び出します。

```
234          addq.l  #8,sp
235
236          neg.w   #B_INCLOSE,d0          *クローズされた?
237          beq     exit                   * そうなら終了する
238
239  LBsRlg  pea.l   eventrec(WORK)         *イベント格納用レコード
240          move.w  #EVENTMASK,-(sp)       *イベントマスク
241          SXCALL  __TSGetEvent           *イベントを取り除く
242          addq.l  #6,sp                  *
243
244  LBress  rts                            *処理完了
245
246  *
247  *       アップデートイベントの処理
248  *
249  update_event
250                                  *自分のウィンドウかどうか調べる
251          move.l  winptr(WORK),a0        *ウィンドウ構造体へのポインタ
252          cmp.l   eventrec+evWhom(WORK),a0
253                                         *自分か?
254          bne     Urets                  * 違う
255                                  *自分のウィンドウだった
256          move.l  a0,-(sp)               *アップデートリージョンで
257          SXCALL  __WMUpdate             * クリップする
258
259          bsr     Pupdate                *画面を描き直す
260
261          SXCALL  __WMUpdtOver           *クリップリージョンを元に戻す
262          SXCALL  __WMDrawGBox           *グローボックスを描く
263          addq.l  #4,sp                  *
264  *
265  Urets   rts                            *処理完了
266
267  *
268  *       アクティベートイベントの処理
269  *
270  activate_event
271                                  *自分のウィンドウかどうか調べる
272          move.l  winptr(WORK),d0        *ウィンドウ構造体へのポインタ
273          beq     Arets                  *0ならどのウィンドウでもない
274          cmp.l   eventrec+evWhom(WORK),d0     *自分か?
275          seq.b   actflag(WORK)          *自分だったらアクティブフラグをON
276                                         *そうでなければOFF
277  Arets   rts                            *処理完了
278
279  *
280  *       タスクマネージャからのイベントの処理
281  *
282  sysmes1_event
283  sysmes2_event
284          move.w  eventrec+evWhat2(WORK),d0
285                                         *d0=イベントの種類
286          cmp.w   #ENDTSK,d0             *タスクの終了?
287          beq     exit                   * そうなら終了する
288          cmp.w   #CLOSEALL,d0           *ウィンドウ全クローズ?
289          beq     exit                   * そうなら終了する
290          cmp.w   #WINDOWSELECT,d0       *ウィンドウのセレクト?
291          bne     Prets
292
293          move.l  winptr(WORK),-(sp)     *自分のウィンドウを
294          SXCALL  __WMSelect             * 一番上にする
295          addq.l  #4,sp                  *
296
297  Prets   rts                            *処理完了
298
299  *
300  *       終了処理
301  *
302  exit
303          move.l  winptr(WORK),d0        *d0=ウィンドウ構造体へのポインタ
304          beq     Pexit                  *0 ウィンドウは未オープン
305
306          move.l  d0,-(sp)               *ウィンドウを破棄する
307          SXCALL  __WMDispose            *
308          addq.l  #4,sp
309
310  Pexit:  clr.l   -(sp)                  *終了コード
311          SXCALL  __TSExit               *タスクを終了する
312
313  *
314  *ここまでが基本部分(どのプログラムでもだいたい同じ)
315  ************************************************************************
316  *以下 プログラム固有のアップデートルーチンと初期化ルーチン
317  *
318
319  *
320  *       画面を(再)描画する
321  *
322  Pupdate
323          move.l  a0,-(sp)               *a0=ウィンドウ構造体へのポインタ
324                                         * =そのグラフポートへのポインタ
325          SXCALL  __GMSetGraph           *自分の描画環境をカレントグラフにする
326          addq.l  #4,sp                  *
327
328          lea.l   testdat1(pc),a1        *テスト用データ
329          bsr     test1                  *16×16ドットフォントで描ふ
330          bsr     test1                  *12×12～
331          bsr     test1                  *24×24～
```

```
332          bsr     test2                  *文字を拡大縮小して描く
333          bsr     test3                  *
334          bsr     test4                  *
335
336          rts                            *描画完了
337
338  test1                          *いろいろな文字飾りをつけて文字列を描いてみる
339          link    a6,#0
340
341          move.w  (a1)+,-(sp)            *フォントの種類をセット
342          SXCALL  __GMFontKind           *
343          move.l  (a1)+,-(sp)            *フォントのサイズをセット
344          SXCALL  __GMFontSize           *
345
346          move.l  (a1)+,-(sp)            *初期ペン座標
347          move.w  (a1)+,d7               *P1=改行幅
348
349          lea.l   testdata(pc),a2        *テスト用データ
350  loop1   SXCALL  __GMMove               *ペン位置を移動して
351          move.w  (a2)+,d0               *
352          bmi     test1exit              *
353
354          move.w  d0,-(sp)               *書体をセットして
355          SXCALL  __GMFontFace           *
356          move.l  (a2)+,-(sp)            *文字列を描画する
357          SXCALL  __GMDrawStr2           *
358          addq.l  #6,sp                  *
359          add.w   d7,pt.Y(sp)            *改行
360          bra     loop1                  *繰り返す
361  test1exit
362          unlk    a6
363          rts
364  *
365  test2                          *文字を縮小・拡大してみる
366          link    a6,#0
367
368          move.w  #0_PLAIN,-(sp)         *書体ノーマル
369          SXCALL  __GMFontFace           *
370
371          move.l  #$0098_0170,-(sp)      *初期座標(4,20P)
372          SXCALL  __GMMove
373          move.l  #$0001_0001,d1         *フォントサイズ増分
374          move.l  #$0008_0008,-(sp)      *初期フォントサイズ
375
376          moveq.l #24-1,d7
377  loop2:  SXCALL  __GMFontSize           *フォントサイズをセット
378
379          move.w  #'あ',-(sp)            *1文字描画
380          SXCALL  __GMDrawChar           *
381          addq.l  #2,sp                  *
382
383          add.l   d1,(sp)                *フォントサイズを増す
384          dbra    d7,loop2
385
386          unlk    a6
387          rts
388  *
389  test3                          *文字を縮小・拡大してみる(その2)
390          link    a6,#0
391
392          move.l  #$0004_0004,d1         *初期フォントサイズ
393          move.l  #$0100_0400,-(sp)      *初期座標(100,4)
394
395          moveq.l #20-1,d7
396  loop3   SXCALL  __GMMove               *ペン位置をセット
397
398          move.l  d1,(sp)                *フォントサイズをセット
399          SXCALL  __GMFontSize           *
400
401          move.w  #'あ',-(sp)            *1文字描画
402          SXCALL  __GMDrawChar           *
403          addq.l  #6,sp                  *
404
405          addq.w  #1,d1                  *フォントサイズを増す
406          add.w   d1,pt.Y(sp)            *描画座標をずらす
407          dbra    d7,loop3
408
409          unlk    a6
410          rts
411  *
412  test4                          *最大サイズで1文字表示してみる
413          link    a6,#0
414
415          move.l  #$0080_0080,-(sp)      *フォントサイズ128×128
416          SXCALL  __GMFontSize           *
417          move.l  #$00dc_00aP,-(sp)      *座標(220,1PP)
418          SXCALL  __GMMove               *
419          move.w  #'あ',-(sp)            *1文字描画
420          SXCALL  __GMDrawChar           *
421
422          unlk    a6
423          rts
424  *
425  testdat1
426          dc.w    0,ROM16
427          dc.w    16,16
428          dc.w    4,8
429          dc.w    20
430
431          dc.w    0,ROM12
```

●アップデートイベント発生時の処理(249～265行)

アップデートイベント時も，まず，そのイベントが自分のウィンドウに向けられたものかどうかを，ウィンドウ構造体へのポインタの比較により判断します。自分だということになれば，アップデート開始です。

アップデートすべきリージョンはウィンドウ構造体のwUpdateに格納されています。このリージョン内だけを再描画すればよいわけですが，わざわざチェックするのも大変です。でも，そこはそれ，うまい方法がちゃんと用意されています。

WMUpdateを呼び出すと，以降の描画範囲がアップデートリージョンに制限されるのです。そのうえで全画面を描き直すと，実際にはアップデートリージョンだけ描き直されます。アップデートが済んだら，WMUpdtOverで，描画範囲の制限を解除しておきます。

ここで，WMUpdtOverのあとでグローボックスを描いていることに注意してください。グローボックスはアップデートリージョン外にあっても描き直さなければならない場合があるので，こういうことをしてみました（もう少しスマートな方法もあるような気はします）。

さて，実際に描画を行うのは322行以下のサブルーチンです。必ず最初に自分のウィンドウ（に対応したグラフ）をカレントグラフにします。カレントグラフは各タスクが個々に設定しますから，これを怠るとほかのウィンドウに落書きしてしまうことになります。

以下，test1～test4の各サブルーチンでは，文字をいろいろな文字飾りで表示したり，拡大・縮小したりして遊んでいます。ここで使っている各SXコールの機能は単純なものばかりですから，説明は省略します。注釈と簡易マニュアルを参照してください。

なお，描画結果を見ると，ゴミが表示されている部分や文字が欠けている部分が見つかるかもしれませんが，それは私のミスでも読者のミスでもありません。

●アクティベートイベント発生時の処理(270～277行)

アクティベートイベント発生時にやるべきことは単純です。イベントレコードに格納された"いまアクティブになるべきウィンドウ"が自分であれば，内部のフラグをONに，そうでなければOFFにするだけです。フラグの操作はseq 1発ですね。

●システムイベント発生時の処理（292〜297行）

システムイベント1と2の処理は，兼用しています。それでかまわないようです。

WINTEST.Xでは，タスクマネージャからのイベントのうち，本当に最小限のもののみをサポートしています。

- タスクを終了しろというイベント
- ウィンドウを閉じろというイベント
- アクティブウィンドウになれというイベント

の3つです。上2つはそのまま終了処理になだれ込めばおしまいです。また，3番目のヤツは，WMSelectで自分をアクティブにすればよいわけです。

●終了時処理（302〜311行）

終了時には，ウィンドウを閉じ，自分で確保したすべてのメモリを解放しなければなりません。

WINTEST.Xでは明示的に確保したメモリはありませんが，WMOpenでウィンドウを開いたときにウィンドウ構造体格納用のメモリを確保していますから，これを解放します。ウィンドウを閉じる処理とメモリを解放する処理を別々に行うこともできますが，ここでは，それらを同時にやってくれるWMDisposeを使っています。

そして，TSExitで終了です。簡易マニュアルによればTSExitで終了コードを返すとあるのでそれに従っています。しかしSX-WINDOW添付のデスクアクセサリのなかには終了コードを返していないものもあるようですから，どれほどの意味があるのかはわかりません。

＊

以上，SX WINDOW上のプログラミングのほんのさわりの部分でした。次号ではC言語で開発する場合について なども解説する予定です。では，また。

<参考文献>

1) Apple Computer「Inside Macintosh Volume I〜V（日本語版）」Addison Wesley
2) スティーブン・チェルーニク「マッキントッシュの道具箱 Vol.1〜3」パーソナルメディア
3) スコット・ナスター「続マッキントッシュの道具箱」パーソナルメディア
4) BNN第二企画部「インサイドマック徹底ガイド 上下巻」BNN
5) 川又ほか「Macintoshの『技術』徹底解剖」インターフェース1988年12月号 CQ出版


```
        dc.w    dd
        dc.w    6,ROM41
        dc.w    24,21
        dc.w    4,1dd
        dc.w    44
*
tsstdeti
        dc.w    d_PLAIN
        dc.l    mes0
        dc.w    d_BOLD
        dc.l    mes1
        dc.w    B_ITALIC
        dc.l    mes2
        dc.w    B_ULINE
        dc.l    mes4
        dc.w    B_ULINE
        dc.l    mes5
        dc.w    d_SHADOW
        dc.l    mes6
        dc.w    B_ITALIC+d_SHADOW
        dc.l    mes6
        dc.w    1
*
mes0    dc.b    '標準(plain)',0
mes1    dc.b    '強調(bold)',0
mes2    dc.b    '斜体(italic)',0
mes3    dc.b    '下線(underline)',0
mes4    dc.b    '抜文字(outline)',0
mes5    dc.b    '影つき(shadow)',0
mes6    dc.b    '組み合わせたりもして',0
        even
*
*       初期化
*
reglist reg     d1-d7/a1-a6
*
init
        movem.l reglist,-(sp)           *とりあえずレジスタを保存する

                                        *コマンド行引数を得る
        addr.w  -(sp)                   *argvは作らない
        addr.l  -(sp)                   *引数文字列も作らない
        pea.l   winsize(WORK)           *ここにウィンドウの位置を得る
        move.l  a4,-(sp)                *a2 コマンド行
        SXCALL  __TSTakeParam
        lea.l   [illegible]
                                        *ウィンドウの表示位置を決める
        btst.l  [illegible]             *……の指定はあったか?
        bne     init4                   *指定があった
                                        *指定がなかったときは
        SXCALL  __TSGetWindowPos        *標示位置を自動に決めてもらう
        move.l  d0,winsize(WORK)        *左上座標をセット
        add.l   [illegible],d0          *それにウィンドウの大きさを足して
        move.l  d0,winsize+4(WORK)      *右下座標をセット
init4   SXCALL  __TSGetId               *自分のタスクIDを得る

                                        *ウィンドウをオープンする
        move.l  d0,-(sp)                *タスクID
        move.w  #TRUE,-(sp)             *クローズボックスあり
        pea.l   -1                      *ウィンドウは一番手前に表示する
        move.w  #W1[illegible],-(sp)    *ウィンドウ定義ID
        move.w  #TRUE,-(sp)             *ウィンドウは可視
        pea.l   wintitle(pc)            *ウィンドウのタイトル
        pea.l   winsize(WORK)           *ウィンドウの位置，大きさ
        clr.l   -(sp)                   *ウィンドウ構造体を確保する
                                        *メモリはメモリマネージャ任せ
        SXCALL  __WMOpen                *オープン
        lea.l   [illegible]
        bmi     initret                 *エラー時は-1でリターン

        move.l  a0,winptr(WORK)         *ウィンドウ構造体へのポインタを
                                        *しまっておく
        move.w  #WINOPT,winoption(a4)   *ウィンドウオプションを設定する

        move.l  a4,-(sp)
        SXCALL  __GMSetGraph            *自分の[illegible]をカレントグラフにし
        SXCALL  __WMDrawGBox            *グローボックスを描く
        addq.l  #4,sp

        moveq.l #0,d0                   *正常終了時は0でリターン

initret movem.l (sp)+,reglist           *レジスタを復帰する
        rts                             *初期化完了
*
wintitle        LASCII  untitled        *ウィンドウのタイトル
        even
*
        .end    exec_entry
```


実践ウィンドウプログラミング

ライフゲームSXLIFE

Nakamori Akira 中森 章

SXシステムの概要がなんとなくわかったところで，ウィンドウ上で動くアプリケーションを実際に作ってみましょう。第1弾はライフゲームです。プログラムは付録ディスクにも収録されています。暁子さんと並べるなりしてご覧ください。

暁子がすき！，それがSX WINDOWを初めて見たときの感想でした。これは犬と自転車に乗った暁子さんがウィンドウ内を動き回るというだけの環境ノフトですが，それだけでSX WINDOWの価値を示すに縦十分なものでした。また，最近オットに流れている環境（？）ノフトにマウスカーソルを追いかけてキョロキョロ動く目玉があります。これも見ていて楽しいアプリケーションのひとつです。

SX WINDOWのようなウィンドウ環境の楽しみ方にはこのような環境ノフトをいくつも並べてその動きを眺めるということがあります。しかし他人が作ったソフトを眺めているだけでは少し物足りません。自分だけの環境ノフトが欲しい。ということで，今回はSX WINDOW上で動く環境ノフトをひとつ作ってみることにしました。ターゲットはライフゲームです。

プログラムの構造

と，元気よくプログラム作成に取り掛かったのはいいのですが，私にはウィンドウ上でのプログラムを作った経験はありません。まったくゼロの状態からの出発です。そこで，私が最初に行ったのはSX WINDOW上で動くプログラムの構成を知るということです。まず，目的のライフゲームに構造が最も近いと思われる暁子.XをDIS.Xで逆アセンブルすることから始めました。このほかにも時計.Xやノート.XなどSX WINDOWについてきたノフトもかたっぱしから逆アセンブルして構造を確かめてみました。

その結果すべてのSX WINDOW上のプログラムは図1のような構造をしていることが判明したのです。つまり，プログラムはCOMMAND.X（やEXECシステムコール）から起動された場合とSX-WINDOWから起動された場合とで異なる入り口を持っており，それぞれ必要な前処理を行います。そして，初期化（ウィンドウのオープンなど）を行ったあとはイベント待ちの無限ループに陥り，発生したイベントに応じた処理を延々と繰り返すというものです（これはMacintoshのプログラムと同じ構造であることがあとでわかったのですが）。

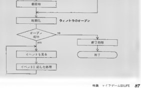
結局これらの処理はそのまま流用すればよく，私たちは残った部分，すなわちプログラムの初期化処理および各イベントの処理を書けばいいでしょう。意外単純だと思いませんか（少し勇気がわいてきた）。

なお，プログラム起動時の各レジスタの値は次のように設定されているようです。

a1	データ領域の先頭アドレス
a2	コマンドラインのアドレス
a3	環境のアドレス
a7(usp)	データ領域の最後＋1
d0	自分のタスクID
d1	複製元のタスクID

ここで，データ領域の先頭がレジスタ（a1）で与えられるところが重要です（データ領域の大きさはプログラムの先頭のヘッダ部分で与える）。すべてのデータ参照をa1レジスタ相対で行って分岐をPC相対で行えば完全にリロケータブルなプログラムを作ることができます（だからどうだといわれてもという利点があるのかよくわからないが，きっといいことがあるに違いない）。今回参考にしたプログラムはどれも（C言語で書かれたもの以外は）a1レジスタをa5レジスタにコピーして，a5レジスタ相対でデータを参照していました。これから作るプログラムもこの作法にしたがっておきましょう。

図1 プログラムの構造



ウィンドウの表示

プログラムの構造が思いのほか簡単で安心したところで次のステップです。ここでは初期化処理を考えます。これはウィンドウをオープンして画面に表示する処理です。これはウィンドウをオープンするためのSX WINDOWのシステムコールである，

WMOpen

を使用すればよいでしょう。付録ディスクのドキュメントによると，__WMOpenというシステムコールは8個の引数を持ちます。各引数の意味は

第1引数	ウィンドウレコードの格納アドレス
第2引数	ウィンドウの位置と大きさ



写真　SXLIFE.X

第3引数　ウィンドウ名へのポインタ
第4引数　ウィンドウ表示の有無
第5引数　ウィンドウの種類
第6引数　生成するウィンドウの
　　　　　前景階層
第7引数　クローズボックスの有無
第8引数　ユーザー定義のデータ

となっています（これもMacintoshとほとんど同じだ）。この引数の値は暁子のノースプログラム（DIS.Xによる）では、

第1引数　disp1(a5)
　　　　　データ領域のどこか
第2引数　disp2(a5)
　　　　　データ領域のどこか
第3引数　disp3(PC)
　　　　　プログラム中のどこか
第4引数　$FFFF
　　　　　可視
第5引数　$320
　　　　　標準ウィンドウ
　　　　　（グラフィックサポート、
　　　　　　クローズボックスのみ）
第6引数　$FFFFFFFF
　　　　　一番手前に表示する
第7引数　$FFFF
　　　　　クローズボックスあり
第8引数　0
　　　　　特に規定しない

となっているので、これから作成するライフゲームでも同じにしておけばよいでしょう（データ参照がa5レジスタ相対になっていますね）。

ただ、ライフゲームではウィンドウの大きさを可変にしない（暁子では固定です）ので、ライフゲームではグローボックスを付けたいと思います。また、ウィンドウに関しても暁子ではグラフィック画面を使う変則的な機能を使用していますから本当の標準ウィンドウにすることにしましょう。このとき、第5引数は208$_H$となります。これは下位4ビット（つまり8$_H$）が「グローボックスのあることを示し、ビット4～11（つまり20$_H$）が標準ウィンドウであることを示しています。

また、ここで注意すべきことは第1引数で示すウィンドウへのポインタが指し示す先は"ウィンドウ"というデータ構造が必要とするだけのバイト数（114バイト）を確保していなければならないということです。もし、第1引数として0が指定された場合はウィンドウのデータ構造のために必要なメモリが勝手に（メモリマネージャによって）確保されるようです。この第1引数に0を与えるかそれ以外のアドレスを与えるかということはウィンドウを消去するときに重要になってきますからしっかりと覚えておきましょう。ここで0を与えた場合はウィンドウの消去に、

　　WMDispose

を使用し、それ以外の場合はウィンドウの消去に、

　　WMClose

を使用するようです。

最初、私は__WMOpenの第1引数に0を指定しているにも関わらず、無条件に__WMCloseを用いてウィンドウを消去していたため、終了時に決まってアドレスエラーが発生してシステムがデッドロックしていました。すべて1に関して参考にしたプログラムの真似をすればよいのに、奇をてらって第1引数に0を指定した（余分なメモリ領域を用意したくなかった）ことによる失敗です。定型的な処理に変な独創性を持たせてはいけないという教訓です。

ところで、このような定型的な操作に従ってウィンドウをオープンしても思ったようにことが運ぶとは限りません。問題はグローボックスです。__WMOpen関数の第5引数でグローボックスを指定してもオープンしたウィンドウには（ウィンドウの大きさを変えることのできるような）グローボックスは表示されません。ただウィンドウの右下に黒い四角が表示されるだけなのです。これにはかなり長い間悩みました。

結局、グローボックスを有するようなアプリケーションソフトを解析した結果、グローボックスを表示するためにはオープンされたウィンドウ（を示す構造体）のwOptionフィールド（先頭から70バイト目）のビット8を1にしなければならないことがわかりました。__WMOpenの返り値（確保したウィンドウ構造体へのポインタ）はレジスタa0に入っていますから、グローボックスを表示させるためにはウィンドウをオープンした直後に、

　　move.w　#$108,70(a0)

という操作が必要になります。このときソースオペランドで指定するイミディエート値の下4ビットは、__WMOpenの第5引数の下4ビットと同じものです。

さて、ウィンドウのオープンについては以上で終わりですが、多くのソフトではこのあとに、

__GMAPage（アクセスするテキスト画面）
__GMForeColor（文字の色）
__GMBackColor（背景の色）
__GMPenMode（文字を書くときの設定）
__GMFontKind（文字フォントの種類）
__GMFontSize（文字フォントの大きさ）

というシステムコールを順番に実行しているようです。この設定はウィンドウをオープンするときの定型処理として覚えておくとよいでしょう。

ところで、__WMOpenではウィンドウをオープンする位置と大きさを指定しなければなりません（第2引数）。この位置は固定値でもよいのですが通常は、

　　__TSGetWindowPos

というシステムコールで次にウィンドウをオープンするのに都合のよいランダムな位置を得るようにします。さらにSX-WINDOWを一度終了して再び立ち上げる場合、前にあったウィンドウの位置を覚えておいてほしい場合があります。このときは、

　　__TSTakeParam

というシステムコールを使えば元の位置を取り出せます。このシステムコールは本来はコマンドラインから立ち上げる場合にコマンドへの引数を取り出すためのものですが、こういう使い道もあるのです。

これまでの説明をまとめると、ウィンドウのオープンを行うための処理は図2のようになります。

最低限の処理

ウィンドウがオープンできるようになったからといって気を抜いてはいけません。プログラムの難しさはこれからです。図1のプログラムの構造でも示したように、次はイベント処理です。この処理をしないとウィンドウをオープンしたあと、システムはウンともスンともいわなくなってしまいます（つまりデッドロックね）。

イベントとはシステムで発生するすべてのできごとのことで、マウスのボタンやキーボードを押したりする動作、あるいはウィンドウの上下関係の変更などがあります。これらのイベントが次々とプログラムに通知されてきますから、各プログラムではそれに応じた処理をしてやらなければなりません。SX-WINDOWで発生するイベントは次の13種類がすべてです。

　　ナイトル
　　マウス左ボタンダウン
　　マウス左ボタンアップ
　　マウス右ボタンダウン
　　マウス右ボタンアップ
　　キーダウン
　　キーアップ
　　アップデート
　　アクティベート
　　ユーザー定義のイベント1
　　ユーザー定義のイベント2
　　ユーザー定義のイベント3
　　ユーザー定義のイベント4

これだけのイベントが発生するわけですが、これらすべてのイベントに対して処理をする必要はありません。プログラムで必要な処理だけを行えばよいのです。

ただ、どうしても必要な処理というものもあります。それはアップデートとアクティベートです。これらのイベントはウィンドウの状態が変化したことをプログラムに知らせるためのものです。ウィンドウの上下関係が変わったときや前のウィンドウに隠れていた部分が現れたときなどにこれらのイベントが発生します。要するにウィンドウ画面の書き直し要求がこれらのイベントなのです。

ウィンドウの書き直しが行われないと悲惨な状況に陥ることは目に見えていますね。これらのイベントに対する処理はそんなに難しくありません。どのプログラムでも同じようなことをやっているだけですから、私たちがプログラムを作るときもそのまま真似をすればよいでしょう（プログラムの具体例は後ほど示します）。

アクティベート、アップデートそれぞれに対して行うべき処理の概要を図3と図4に示します。図3のアクティベート処理に関してはアプリケーションに関わってくる処理は何ひとつありませんから、どんなプログラムでも同じことをしておけばよいと思います。図4のアップデートに関しても実際に画面を書き直す以外の処理はまったくそのままでかまわないでしょう。

さて、ここまでの処理に関して2つほど注意すべきことがあります。まず、自分のウィンドウに対して何か処理をする場合には必ず、

　　__GMSetGraph

というシステムコールを実行しなければなりません。これは、早い話が、これからウ

図2　ウィンドウのオープン

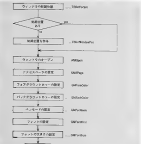
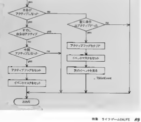


図3　アクティベート処理



ィンドウの内容（画面）を変更するよという宣言みたいなものです。ウィンドウを参照する場合のオマジナイとして覚えておきましょう。SX-WINDOWのプログラム（もうすぐ紹介しますからね）をよく注意して眺めると至るところでこのシステムコールが使われているのがわかると思います。その次は背景の色です。これはウィンドウをオープンした直後

　　GMBackColor

というシステムコールで設定しますが、これは色番号3（明るい灰色）に設定しておくのが一番楽です。この色に設定しておくと、自分のウィンドウがアクティベートされ、他のウィンドウで隠されていた部分が見えるようになった場合でも、アップデート処理で背景を描き直す必要がありません。他のウィンドウで隠されていた部分がたまたまこの色になっているからです。これを利用するとアップデート処理を高速に行うことができるのです。

ところで、実際のプログラムでは、これまで説明してきたアクティベート処理やアップデート処理のほかに、もうひとつ重要なイベント処理が必要になります。それはマウスボタンが押されたときの処理です。

確かにアクティベート処理とアップデート処理さえ書いておけばちゃんとウィンドウは表示され、他のウィンドウと仲よくマルチタスクしながら画面上に存在することができます。でも、それだけです。表示されたウィンドウは動かすことも消し去ることもできません。ウィンドウを移動したり消し去るためにはマウスの左ボタンを使うのが慣例になっていますから、マウス左ボタンダウンのイベント処理としてそれらの処理を書いておかなければウィンドウとしては不十分なのです。

このマウス左ボタンダウンの処理にはそれぞれのプログラムでいろいろなバリエーションがあり、こうするのがいいといった定型的な処理はありません。ここではウィンドウマネージャ（ウィンドウマン）を使用した方法を紹介しておきます。これはキャンバスXの中で使われている方法です。ウィンドウに対する移動とか拡大とかの処理は、本来ならそれぞれ専用のシステムコールを呼び出して処理しなければなりません。しかし、SX-WINDOWでは発生したイベントと対象となるウィンドウへのポインタを、

　　__SXCallWindM

というシステムコールに渡せば自動的にその処理を行ってくれます。すごく便利な機能なので、ありがたく使わせてもらうことにします。ただしクリッピングの更新は

　　__GMClipRect

の実行が必要です。

ウィンドウマネージャを使用したマウス左ボタンダウンの処理を図5に示しておきます。この処理を簡単に説明すると、マウスがクリックされた位置がクローズボックス（ウィンドウの右上についている×の部分）の中であれば終了処理をしますが、それ以外はすべてウィンドウマネージャに任せるということになります。前にも説明したとおり、終了処理でのシステムコールは__WMCloseと__WMDisposeの区別をしっかりとしなければなりません。

さて、これまでのまとめとして、ここで簡単なプログラムを紹介しておきましょう。リスト1がそのプログラムですが、これはSX-WINDOWの画面上にのっぺらぼうのウィンドウを表示するプログラムです。一応、アクティベート、アップデート、マウス左ボタンダウンといった最低限のイベント処理は書いてありますから、ウィンドウの移動、拡大、消去は自由自在です。ここまでくれば目的（ライフゲームですよ、忘れないでね）は8割方達成したも同然です。

図4　アップデート処理



ライフゲーム本体

ウィンドウ表示のほうはなんとかメドがつきましたから、本体であるライフゲームについて考えましょう（ライフゲーム自体については参考文献を見てね）。SX-WINDOW上でのライフゲームの基本的な動作は、ある初期状態から始めて、1単位時間経過することの状態を次々にウィンドウに表示することになります。

このウィンドウ画面の変更はシステムになにも他のイベントが発生していないとき、すなわちアイドルイベントが発生したとき行います（アップデートイベント発生時にはもちろん画面を書き直す必要がある）。

したがって、このライフゲームでは時分割でウィンドウに与えられてくるアイドルイベントのわずかな時間内でウィンドウ画面の書き換えを行わなければならないため、かなりの高速性が要求されます。とはいえ、1画面の状態が変更されるのを待っていたのでは（10MHzの68000では）どうしても処理が遅くなってしまいます（このとき他のウィンドウの動きも遅くなってしまう）。

そこで考えたのが1回のアイドルイベントにつきウィンドウ画面の1行のみを変更するということです。これによって他のウィンドウの動きに影響を与えることなく、そこそこに高速なウィンドウ画面の書き換えが実現できます（できる予定です）。

それでは今回のライフゲームの1行書き換えアルゴリズムを実現するC言語のプログラムをリスト2（1行の書き換え）、リスト3（アップデート処理時の全画面書き換え）、リスト4（ライフゲームの初期化処理）に示しておきましょう。リスト2において、fieldという2次元配列が存在/非存在の状態を保持するセルの集まりで、lifelという関数が1回の呼び出しごとに引数で与えられたfieldの特定の1行が次の時刻にどのように変化するかを調べています。そして変化のあったセルだけをウィンドウ画面上で書き直すようにしてさらに速さを稼ぐようにしています（実際に書き直すのは引数で与えられたよりもひとつ前の行ですが、なぜそうしなければいけないか考えてみましょう）。

それというまでもなく、psetがウィンドウ画面上に点を打つための関数、presetはウィンドウ画面上の点を消すための関数です。リスト3、リスト4については説明は不要でしょう。というわけで、リスト2、リスト3、リスト4を先に作ったウィンドウ表示のプログラムに組み込めばライフゲームの完成です。

実際の組み込み作業はリスト2、リスト3、リスト4のプログラムをGCCでコンパイルし、出てきたアセンブリ言語ソースを手で変更することで行いました。すべてのデータ参照をa5レジスタ相対に変えてしまったので、コンパイルされた直後のソースリストの面影はあまり残っていません。ただ、リスト3についてはあまりに遅そうだったので超高速で画面の書き直しができるようにアセンブリ言語で1から書き直してしまいました。

このときも背景色が明るい灰色であることを利用して存在点のみを書き直すだけにしています。それ以外に関してはアルゴリズムが複雑であることもあり、特にアセンブリ言語で書き直すことはやっていません。

ところで、pset関数やpreset関数には、

　　__GMPutRImg

というシステムコールを使用しています。最初は＊とか■といった文字をウィンドウに表示することで存在点を表そうと思ったのですが、いろいろと試行するうちに一度ウィンドウに書き込んだ文字を消去するのはかなり面倒だということがわかりました。それで結局パターンをそのまま書き込むことにしたのです。これによって表示するパターンをいろいろとデザインすることができるようになり、このほうがかえってよかったかなと思っています。

パターンの設定機能

ここで完成したライフゲームの全リストをリスト5に示します。このリストでは今まで述べた機能のほかにパターン（ゲーム）の設定機能を持たせてあります。パターンの設定はマウスの右ボタンで行います。ライフゲームのウィンドウ内でマウスの右ボタンが押されるとウィンドウ画面の更新はロックされ（userWKKという変数をフラグに使っている）パターンの設定モードに移ります。あとは、

　　__EMMSLoc

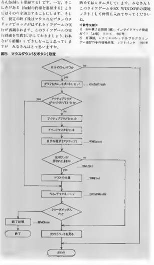
というシステムコールを使ってマウスの位置を知り、その位置に点を描くだけ（もちろんfieldにも登録する）です。一方、そこに点がある（fieldの内容を参照する）ときにはその点を消去することにします。そして、設定の終了後はマウスの左ボタンのクリックでロックがはずれライフゲームの実行が再開されます。このライフゲームの実行画面を写真1に示しておきましょう。われながら結構いいできになったと思っていますが、みなさんはどう思いますか。

＊

なにもかも手探りの状態で作り始めたSX-WINDOW上のライフゲームですが、既存のプログラムの手法を真似ることにより一応の完成を見ました。一見不可能に思えることもやればなんとかなるものなのですねえ。私は毎日のようにライフゲームを眺めてはニタニタしています。みなさんもこのライフゲームをSX-WINDOWの環境ソフトとして仲間に入れてやってくださいね。

＜参考文献＞
1) BNN第2企画部（編）、インサイドマック徹底ガイド（上巻）、BNN、1987年
2) 有澤誠、レクリエーショナルプログラミング　遊びの中の情報処理、ソフトバンク、[illegible]年

図5　マウスダウン（左ボタン）処理



（注意）本誌に掲載したリストにはウィンドウズーム時のクリッピング処理が抜けています（ぼーっとして気がつかなかったんだもん）。でも、付録ディスクのリストでは直してあるのでご安心を

リスト1



▶うーむ　クリスタルボルシェ……なんて高価な消しゴム（違うって）。
中京市　雅之(23)雑記帳

リスト2

リスト3

リスト4

リスト5

▶RPGを作ろうと思った。それで資源を作って終わりました。ふと、シューティングが作りたくなった。そして、眠くなった。結局なにもできなかった。ソコン歴3カ月の私……。
日立　大輔(17)高校生

謹賀新年 PRO-68K

Oh!X史上2度目の付録ディスクをお届けしよう。ちなみに左が今回のディスクの内容を一覧としたものだ。

今回のディスクにはいくつかの目標があった。そのうち、ひとつはシステムを入れ、起動可能にすること。ディスク容量を削ることになっても、だ。さらに、ソースプログラムをできるだけ掲載すること。できるだけ機能プログラムを入れないこと。できるだけ使いやすくすること……これらがどの程度達成されたか、その目標が正しい目標であったかについて、その評価はこのディスクを手にしたあなたに委ねることにする。

ディスク容量の関係から削除せざるをえなかったものもある。時間的に十分でなかった面もある。それでも、「ディスクでしかできないこと」を、提供したと信じたい。

CONTENTS

付録ディスク利用の手引き

編集部

お待たせしました。ついにディスク第2弾です。多くの方から期待のお手紙をいただきました。ありがとうございます。今回は前回のディスクでの反省点もふまえて、初心者にもわかりやすい構成にしてみました。ではまず付録ディスクの取り扱い方　圧縮されたファイルの取り出し方などについて解説しましょう。なお　起動時はRAMディスクがAドライブに設定されていますので注意してください。

みなさま、あけましておめでとうございます。

予告どおり今月はディスクつきです。1990年6月のような大物はありませんが　それとはまた違ったアプローチからディスクを構成してみました。

ファイルの圧縮には LH X ver 1.01m を使用しています。これは吉崎栄泰氏の原作、山本浩一氏の移植によるものです。ディレクトリごとの圧縮など　今回のディスク作成では大活躍しています。そのため、「やったー！　1月号は特別付録だー、生ディスク3枚用意して待ってますよん」という宮城県の相沢さんには申しわけないのですが、生ディスクは4枚必要となってしまいました。

特徴

今回のディスクおよびそれを展開したもののすべてがシステムつきですのでそのまま起動可能です。

X68000は発売当初よりビジュアルシェルが搭載されていたことはご存じのとおりですが　クオリティグラフィックやマルチメディア　高機能など真の意味での汎用性を志向したユーザーコンセプトからは私たち自身もの足りなさを禁じ得なかったことも事実です。そこでVS2.Xを開発しました。流通する多くのコンピュータが16色表示という環境のなかで　高解像度65536色グラフィックを実現しています。



1.起動のようす

今回のディスクはすべてVS2.X上から操作します。

注意

ディスク内のほとんどのファイルは圧縮されてます。使用する際には所定の解凍作業が必要です。受験生は自己の責任において解凍するようにしてください。

また SX WINDOW 関係はシャープ提供の開発者向け資料を編集部で独自にまとめたものです。こういったものは使の常として　使用法や内容に関する質問をシャープに問い合わせることは絶対にしないでください。編集部に問い合わせることもご遠慮ください。質問されても回答できません。

新年早々にもソフトバンクからSX WINDOWの解析本が発売される予定ですので　それらを参考にしてください。どうしてもわからない部分はたぶんこちらでもわかりません。サンプルプログラムやシステムプログラムを読んでください。コードがすべてを語ってくれるはずです。

また「Oh!Xでは半年に1回ディスクをつけるんだ」と早合点しないように。

ディスクの使い方

具体的に付録ディスクを起動してからの作業を列記してみましょう。

1）プロテクトシールを貼る

確性面には貼らないでください。



これが起動画面

2）ディスクを起動する

起動すると階下に延々と必要なファイルを展開します。とりあえず静観してください。やがてビジュアルシェルが立ち上がります。起動時はできるだけソフトキーを押しながらリセットするようにしてください（RAMディスクの初期化）。うまくいかない場合は電源を落とします。X68000のRAMはタフですから電源を落として10秒以上はっておいてください。

3）「はじめに.DOC」のアイコンをダブルクリック

説明が出ます。

4）新しいディスクをドライブ1に入れる

必要ならフォーマットします。

5）バックアップを取る

展開作業に入る前にディスクをバックアップしてください。Bドライブのディスクアイコンをつかまえて C ディスクのところに重ねます。完璧に重ねようとすると失敗します。あくまでもマウスカーソルの先がCドライブのアイコンを指すように重ねてください。

6）バックアップディスクを取り出す

Cドライブのアイコン上で右クリックしメニューから「close/eject」を選んで排出します。

7）新しいディスクを入れる

必要ならフォーマットし、すでになにか入っているディスクの場合もディスクアイコンのところからフォーマットを選んでください。

8）展開作業に入る

DISK1～4.BATをダブルクリックします。なお　展開したディスクはすべてIPL起動可能です。

問題点

例によって、現在までに判明した今回のディスクの問題点を挙げておきましょう。

■1Mバイトでは動かないものがある

Michael はビジュアルシェルからでは起動できません。Zs EX はできれば4Mバイト以上で使ってください。

■Zs EX のアイコンが違う

使う分には関係ないのですが、MAP時のアイコンはZ'sTRIPHONYのものです。新たに作り直したのですが、手違いで開発中の暫定版（？）が入ってしまいました。

■ディスク4のREADME.DOCの内容が違う

容量の都合で一部のサンプルはディスク3に移動しました。とけいXは収録されていません。理由は聞かないでください。

■SXLIFE.Xが全クローズで消えない

個別に消してください。

■ボリューム名をつけそこなっている

最後につけた部分でした。

■一部のドキュメントに誤字がある

余力があれば直す予定でした。

＊　＊　＊

細かい部分は解凍したディスクを実際に起動して確認してみてください。

謹賀新年PRO-68Kを使うには，あるいはX68000 SUPERユーザーに贈るビジュアルシェルの使い方

Nakano Shuichi
中野 修一

発売当初のX68000を輝かに印象づけたもののひとつはいまだ色褪せない名作「グラディウス」、そしていまひとつはビジュアルシェルことVS.Xでしょう。

さて、今回のオマケディスクはビジュアルシェル上で動作します。前回手厚らがあった展開作業も、ディスクさえ入れてやれば、あとはアイコンをダブルクリックするだけで自動的に完了します。

Human68kやCOMMAND.Xのバージョンの違いによる誤動作を防ぐため　最新のHuman68k ver 2.02を搭載しました。これで初期型X68000を買ってよ年前ver 1.0のHuman68kしか使っていなかった方も安心です。

さらに「OPMDRV.Xはどうやって手に入れるんですか？」という方のために　強力な実績を持るOPMDRV.X ver 1.01を　そのほか　変貌な回数回を除いてプログラムの動作に必要なデバイストライバをすべて搭載しました。

そして、「せっかくのディスクをフォーマットしちゃったんですが……」という悲劇を繰り返さないためにも　誰でもわかるビジュアルシェルが搭載されたわけです。

一説によると「ユーザーフレンドリ」ともいいますが「全部面倒みなけりゃ、どうせなにもできねえだろ」ということですから、規範を継のためUNIXの得意としては増殖しされていると慎重するほうが健全です

当然SX-WINDOWが注目されるところですが　メインメモリ1Mバイトで動作させるため、VS.Xが日の目を見ることになったのです。ディスクに入っているVS.Xはまったく従来どおりのものですがVS.XはX68000SUPERなどのSCSIデバイスに対応していないなどの問題があるため　独自に改造を加えたVS2.Xも用意しました。

今後の動向は流動的ですが　少なくとも次にディスクをつけるときにも同様の構成をとることになるでしょう。特にX68000SUPERユーザーの方　ビジュアルシェルの操作に慣れておいてください。

ビジュアルシェルの基本

まず　基本操作から

■ディスクのフォーマット

生ディスクを挿入すると　フォーマットするかどうか聞いてきます

■ディレクトリの移動

ディレクトリアイコンをダブルクリックすればそのディレクトリのウィンドウが開きます。通常はQUICKSTARTのディレクトリが最初に開きますので　ルートに移りたいときはドライブアイコンをダブルクリックします

●ファイルのコピー

SX-WINDOWと違って、ウィンドウ上のアイコン移動はすべてムーブ（移すると元の場所からなくなる）となるので　コピーしたい場合は一度コピー機のアイコンのところまで持っていってください　コピー機から呼び出したアイコンをつかんで持っていけばコピーが実行されます。

■ディスクのイジェクト

ディスクアイコンのところで右クリックしイジェクトを選んでください。

●ディスクのコピー

ディスクアイコンをひっつかまえて　目的のディスクの上に重ねてください。ハードディスクとフロッピーディスクなど、違う種類のメディアはディスクコピーできません　同じメディアでも容量が違うとディスクコピーできません。

フロッピーディスクがEドライブ以降に割り当てられているときは　画面上に1個のディスクアイコンしか出せません。そんなときはいったんコピー機のところへアイコンを置いてからディスクを取り換えてください。

●終了

なにも選んでない状態で右クリック　QUITを選択します。

応用

ここまでは基本。単に付録ディスクを使うためならこれで十分です。通常のシステムディスクに入っている全然使いこなされていないVS.Xを見てビジュアルシェルはなにもできないと思っている人がいるともったいないのでもう少し解説します。

さらに使いこなすためのキーとなるのはアイコンです。

アイコンはただ並んでいるだけではありません。アイコンのそれぞれはディスク上のプログラム　データを表します。それぞれなんらかの規約のもとに作成されたもので　呼び出されれば所定の機能を果たします。普段は小さくて分散なときに大きくなるカプセル錠剤のようなものです。

そして　ダブルクリックはその割りを溶ます。では　割りから出たファイルはどうなるべきでしょうか？

通常　さまざまなファイル　BASICのテキストファイルとか　文書ファイル、音楽データなど……プログラムでは違うものとして扱いますが　ディスク上ではなんの違いもないデジタル情報、ビット列にすぎません。これらが本来どのように扱われるべきかというのは、決まっているようで決まってないのです。

もしかして　＊.X形式の実行ファイルを65536色のグラフィックと見なして表示したいとか　グラフィックツールで描いた絵をADPCMで鳴らしたい――と思う方もいるかもしれません　可能です。データがどのようにふるまうべきかというのはユーザーがすべて設定できるのです（対応するプログラムがあれば、だが）。

これを設定するのがアイコンメンテナンスです。実行ファイルとパラメータを指定することができます。起動したファイル名は自動的に渡されますから

起動ファイル
ファイル名
パラメータ

と　ひととおりの指定が可能です。付録ディスクではおおまかな指定がしてありますが　コマンドシェルが使える人なら独自のシステムを作ることも可能でしょう。

通常の実行ファイルは無指定でも　そのまま実行されます。データファイルの場合はなんらかの実行ファイルを指定します。なにも指定しないと「ファイルは実行できません」エラーになります（VS2はこの部分を取り出して拡張している）。

このとき、起動ファイルのパス指定はどうすればよいのでしょうか。基本はあくまでフルパス指定です。が　ちゃんとそのアイコンがあった場所がカレントになりますのでドライブを変更したりするとデータファイルの位置に依存した指定となります。

ですから　おまけディスクの＊.PICのファイルはアイコンのオプションを

=¥BIN¥VSPIC %

から

=A:¥BIN¥VSPIC %

に変更しましょう。

＊　＊　＊

時代はもうSX-WINDOWに向かって流れているのかもしれません　そのSX-WINDOWは当然VS.Xを超えるものでなければなりません。しかし　此我は真なんの進化もさせてもらえなかったVS.Xを超えてもあまり意味はないでしょう。VS.Xの延長で可能であっただろう部分を雲の　真っ向から超えていってほしい　そう思います

VS2.Xの使い方

Kageyama Hiroaki
影山 裕昭

ビジュアルシェルの基本的な使い方がわかったところで、今回拡張された部分について見ていきましょう。シングルウィンドウと融通は利きませんが、VS2.Xはもっとも簡単な方法で最大限の効果を上げています。皆さんのご意見をお聞かせください。

これから紹介するVS2.XはVS（ビジュアルシェル）にちょっとばかし便利な機能を追加するプログラムです。このプログラムはOh!Xのオマケディスク用のシェルとして企画/開発されました。

VS2.Xの起動はコマンドラインから

A>VS2

で行います。パスの通ったディレクトリにバージョン1.12のVS.Xがあると、これから紹介する拡張機能が使えるようになります。追加された機能は

ファイルの内容表示　>TYPE
ファイルのコピー　>COPY
外部ファイルの実行　=

の3つです。これらはすべてアイコンメンテナンスのパラメータで設定します。


普通 ファイルを開き


80桁で表示した例

ファイル内容の表示

VSでは*.Xファイルをダブルクリックで実行することができました。どうせならダブルクリックでファイル内容の確認もできればもっと便利だろうということで作ったのが>TYPEコマンドです。ここでは拡張子がDOCのファイルをダブルクリックしたときに、ファイル内容を表示するようにしてみましょう。

まず最初にVSのアイコンメンテナンスを選択します（取扱説明書2.5デスクアクセサリ「アイコンの(8)アイコンメンテナンスの使い方」を参照してください）。ウィンドウが開いたら、

アイコン名　*.DOC
実行ファイル
パラメータ　>TYPE %

と書きます。コマンドは大文字でも小文字でも構いませんが、すべて半角文字で書いてください（以下紹介するコマンドはすべて半角で書きます）。これで準備完了です。アイコンメンテナンスを終了して、拡張子がDOCのファイルをダブルクリックしてみてください。普通なら「このファイルは実行できません」となるのですが、パラメータに>TYPE %を指定してあるので、ウィンドウがオープンしてファイル内容が表示されます。ここでのキー操作をまとめておきます。


カットファイルも表示できる

F1　先頭行に移動
F2　最終行に移動
↓　正スクロール
↑　逆スクロール

ほかにロールアップ/ロールダウンが使えます。また電脳倶楽部のようにマウスの左クリックで正スクロール、右クリックで逆スクロール、両クリックでウィンドウをクローズすることができます。画面上ではウィンドウの左上のマークをクリックするとウィンドウをクローズします。また右上のマークをクリックするとウィンドウを大きく開きます。いいとれましたか。初めからウィンドウを大きく開きたいときはパラメータに

>TYPE -80 %

と指定してください。80は80桁という意味です（将来は可変にするつもりだが、現バージョンでは80以外はエラーとして扱っています）。これはトグルスイッチになっているので、続けてクリックすると元の大きさに戻り、さらにクリックするとまた大きく……とウィンドウのサイズを交互に変更することができます。

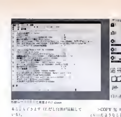
さらに、上のタイプコマンドでは電脳倶楽部でお馴染みのカットファイルを表示することもできます（ただし白黒が反転している）。

VS2上のプログラム

VS2はマルチタスクでもマルチウィンドウでもない。基本的にポップアップウィンドウひとつしか扱わない。わざわざVS2のうえでプログラムを走らせるより、ふつうに起動したほうが面倒がない。あえてやっている理由は「雰囲気がよい」の一点に集約される。

よって、VS2上のプログラムはもっぱらウィンドウの中で動いている「ふり」をすることが要求される。その代わり、ハードウェア資源のうちスプライトとテキストプレーンの1を除いたすべてが使える。プログラム作成はとても楽である。

今後の方向として、常駐終了させてタスクを切り替えたり、コンソールを開いたりといったことも理論上は不可能ではないが、プログラムの性格上、1Mバイトで動いて十分に小さいものが要求されているのであまり期待はしないように。


外部コマンドで用意されたcbase


PICファイルの表示例

ファイルのコピー

これまで本誌の音楽特集などでVSからダブルクリックでOPMファイルを演奏する方法を紹介してきました。あれを試した人はわかると思いますが、見ていてあまり美しいものではありません。それはファイルのコピーにCOMMAND.Xを使う必要があったからです。そこでコピーコマンドを追加しました。コマンドは>TYPEと同じようにパラメータに指定します。

アイコン名　*.OPM
実行ファイル
パラメータ　>COPY % OPM

とパラメータに書いておくだけで、VSを抜けずにOPMドライバにファイルをコピーできるようになります。もちろんOPMドライバが登録されていなければ演奏されません（あたりまえですね。未登録の場合はカレントドライブにOPMという名前のファイルが作られます）。

ほかにも

>COPY % PCM
>COPY % IOCS

といったようなこともできます。各自試してみてください。

外部コマンドの実行

最後に紹介する=コマンドは外部プログラムの実行をするものです。普通の実行と違い、VSの画面が表示された状態のままプログラムが実行されます。ですから指定する外部プログラムはVS用に作られたものでないと、実行終了後に画面が崩れることもあります。

今回はPICファイルを表示するVSPIC.Xを丹氏が、カード型データベースCBASE.Xを泉氏が作成しています。

CBASE.Xの解説は泉氏の記事を読んでもらうとして、VSPIC.Xについて触れておきましょう。付録ディスクにはVSPIC.Xが収録されています。このプログラムを使えばVSの画面上でPICファイルを見ることができます。

アイコン名　*.PIC
実行ファイル
パラメータ　=A:¥BIN¥VSPIC %

のようにします。実行ファイルにはなにも指定せず、パラメータで=に続けて実行ファイルを指定してください。設定すればメーカー標準のGL3ファイルもロードできますが、今回のディスクに収録されているアイコン設定ではサポートしていません。

また、左記のように実行ファイルは基本的にフルパスで指定してください。表示された画像を消すときはマウスを両方クリックするか、ESCを押しましょう。

最後に

今回収録したVS2.Xは改善の余地がかなりあり、自分自身では不満な点も多くあります。開発期間が短く、急いで作って中途半端なものにしたくなかったので、今回はあえて評価版というかたちにさせてもらいました。ご意見、ご要望があれば編集部までお寄せください。

次の機会にはバージョンアップしたものを紹介する予定です。

VS2.Xの高速化

今回のディスクではIOCS.Xを組み込んであるので、アイコンの表示などが高速化されているが、それ以上の高速化の手段もないではない。

満開製作所発行の電脳倶楽部vol.21で発表されたMOUSE.Xを組み込むとマウスが軽くなる。特にマウスカーソルのアニメーションなどが高速化され、結果としてウィンドウのクローズなどが速くなる。

また、通常CONFIG.SYSで設定されている「VERIFY ON」の指定をOFFに変えるとディスク操作が大幅に高速化される。これはディスクのエラーをチェックする部分なので、常識的にいって、不用意におすすめすべきものではない。これまでVERIFY OFFで作業している人は少ないと思う。いままで使っていて、ディスク速度で不満を持ったことがないならあえてOFFにする必要はない、ともいえる。逆に、これまでディスクエラーなど出たことがないし……という場合ならOFFにしといてもいいんじゃないか、ともいえる。各自の責任において判断してほしい。

そのほか、TYPEのスクロールは山ほどウエイトが入っているので好みで調整すること。

VSPIC

柳沢明氏によるPIC.Rのルーチンを使ってビジュアルシェル上に画像を表示するもの。すでにご存じの方も多いと思うが、X68000は768×512モードでも制限付きながら65536色表示することが可能である。これはビデオコントローラを直接操作することにより実現される。

問題となる制限は「512×512ドット分しか表示できない」というものだが、今回は512ドット分以外のところをマスクすることで、見かけ上グラフィックウィンドウを確保している。もともとVS2はシングルウィンドウのシステムだから特に問題はない。

このモードではドット比が1：1に近く、また、画質も良好である。大きさを除けばX68000でもっともグラフィックの美しいモードといえる。今回はPICファイル用にドット比を補正して表示している。よほどのことがないかぎり、不自然さはないと思う。

システムおよびツール類

編集部

今回は都合により誌面で十分なページを割いて解説できないプログラム，データ群が出てきてしまいました。ものによっては来月号以降で補足する予定です。とりあえずひととおりのツール群について最低限の解説をまとめて行っておきましょう。

CBASE.X (DISK1)

X-BASIC開環実例で泉人介氏が作成したカード型データベースをC言語にコンバートしてチューンアップしたもの。VS2.X上で動作する2番目のプログラムだ(1番目はVSPIC.X)。といっても構造はふつうのプログラムと変わりない。11画面表示部分を変更，呼び出され方でCOMMAND.Xからでも VS2.Xからでも同様に動作する。

テキストデータの取り込み，書き出しや複数項の一括検索など，実用に耐える仕様に強化されている。扱えるデータ数はまだ十分とはいえないが，小規模の用途でなら十分に使える。

詳しくは今月のX-BASIC開環実例を参照のこと。

INDEX90 (DISK1)

昨年(1990年)1年間のOh!X掲載記事をカード型データベースに入力したもののはずだったが，残念ながら，データ入力は不完全である。特集，連載/シリーズもの関係はほぼ実用レベルのデータが揃っているが，単発記事は全滅に近い。データを補完するもよし，CBASE.X用のサンプルデータとして使ってほしい。

DICTOOL (DISK3)

1990年9，10月号で村田敏幸氏が発表したASK68K用の辞書メンテナンスツール群。辞書ファイルから特定の語調だけテキストデータにしたり，編集したテキストデータから新しい辞書を構成したり，X68000やPC-9801用に発売されている日本語フロントプロセッサ用の辞書をASK68K用にコンバートしたり，といった日本語入力環境を改善するためのもの。

簡単な使用法ならば，それぞれのプログラムをパラメータなしで起動するとヘルプメッセージとして表示される。さらに詳しい使い方は掲載号の該当記事を参照すること。

FONT (DISK1)

IOCS.Xのフォント切り替えのサンプルとして，過去に電脳倶楽部に掲載された平木敬太郎氏の作成による8×16ドットの半角英数字フォントを収録したもの。

IOCSのフォント切り替えはIOCS.XがCONFIG.SYSによってデバイスドライバとして組み込んであることが前提条件となる。

方法はコマンドラインからは

A>COPY NAMI.FON @IOCS

のように，システムファイル名"@IOCS"にデータをコピーするだけだ。付録ディスクによって作成されたVS2.X上でなら，フォントファイルのアイコンをダブルクリックするだけでよい。今回は変更用のフォントしか用意していないので，切り替えたフォントを元に戻すにはリセットするしかない(IOCS.Xを解除すれば別だが)。注意してほしい。



切り替わったフォント

HASH.X (DISK2)

一算学氏による半影つきレイトレーサ。通常，レイトレーシングを始め，コンピュータグラフィックスでは難しいとされる柔らかな影を実現したレンダリングシステムだ。これは，HAlf SHadow ray tracingシステム，HASHというプログラム名でディスクに収録されている。

残念ながら，アルゴリズムなどの詳しい解説は来月に譲ることにして，ここでは基本的な使い方とサンプルについて解説する。

まず，使用法。コマンドラインからは

A>HASH データファイル

で起動する。途中で終了チェックをしていないので，どうしても計算を途中でやめたいときはインタラプト+Aを押す。VS2.Xからは，単にデータファイルのアイコンをクリックするだけでとりあえず実行は始まる。

データファイルはED.Xなどのテキストエディタで作成する。

このシステムで使える物体は球のみで論理演算はない。レイトレーシングの経験がある人ならば，データファイルのサンプルを見るだけで，独自のデータファイルを作成できるだろう。

最初にあるS項はシステム設定で画面の解像度や色数，ディザ，表示範囲，解像度，反射回数，半影をつける際の解像度を表すそのほか視点設定，光源設定，アトリビュート設定，背景設定などは目数を見ればだいたいの見当がつくはずだ。



CBASE.X





RING



BALLS

●PYRAMID

光源は，画面左上に見えている半形波光源とピラミッドの頂点の線状光源，それに視点の無限後方からピラミッドを照らすものの3つ。計算時間は mesh=40で10.37時間，mesh=1(通常の影処理)で6.87時間。この場合，通常の影処理の場合に比べて1.6倍程度の時間で半影を計算している。なお，ここに記す計算時間はGNU-C，FLOAT3.X，数値演算プロセッサボード→256×256ピクセルによるものであるが，計測によるとFLOAT2.X でも計算時間はさほど変わらないようである。原因は半影処理にほとんど実数計算を使わなかったことにあるとも思えるが，数値演算プロセッサが遅すぎるものではないだろうか。

●BALLS

4つの球を正四面体の頂点に配置。白い球が光源で，非常に半径が大きい。その他の3つの球のぼやけた影をよく地面に落としている。球の向こう側に見えるシャープな影は，見掛けの大きさが非常に小さな無限遠光源による影である。計算時間は，mesh=40で3.16時間。

●RING

HSV色座標系で求めた10色の球を円形に配置。地面に一部だけ膜を張した球体に映り込んでいる。地面は，白と緑のチェック模様。計算時間は，mesh=20，85×85ピクセルで1分。

HASHのデータサンプル

```
[illegible]
```

●　●　●

理屈もわからず操作するだけではつまらないかもしれないが、まずはこのシステムが作り出す独特の映像を鑑賞してほしい。詳しくは来月号で。

MUSICDRV.ver2.0

サン・ユニバーサル・サービス提供のMUSICDRV.Xのバージョンアップ版。FM音源部の演奏がかなりOPMDRV.Xに近くなった。これまで仕様の違った「Q」の動作が同じになったことと、OPMDRV.Xコマンドを素通しするモードの追加により、これまでのBASICに比べてもほとんど違和感のないコーディングが可能。MIDI部分の高性能はそのままに、和音関係はかのバグもなくなり、MIDI制御には絶対の強さを誇る。

おまけにAD PCM関係ではOPMDコンパチのPCMデータを受け付ける。加えてOPMD用のAD PCMコンフィギュレーションがある場合はそれがそのまま使用できる。これまで

A>OPMD BOS.CNF

とやっていた方なら、

A>COPY BOS.CNF MIDI

とすればよいわけだ(ただしPCMバッファは拡大すること)。

まだ、ときおりエラーを出しておかしな演奏になることもあるが、エラー箇所は的確に表示されるので対処はさほど難しくないだろう。



MUSICDRVの組み込み

MUSICDRV.Xの特徴

さて、MUSICDRV.Xといっても全然わからない方もいると思う。ここでその特徴をひととおりながめてみよう。

●MIDI+FM音源+AD PCM

まず、音源ならなんにでも対応している。MIDI16チャンネル、FM音源8チャンネル、AD PCM 1チャンネルが同時に演奏可能。

●和音指定

16チャンネルのシーケンスができる。とはいっても、本来1チャンネルで多くの音を出力できるMIDI楽器に比べ、OPMDRVなどは1トラック1音が基本となってる(FM音源なら当然のことだが)。このままでは和音が出てきたときにひどく無駄な処理をしなければならないし、同時発声16音ではMIDIが本来持つ表現力を生かし切れないことになる。MUSICDRV.Xでは1トラックでの和音記述を可能としたことにより合理的なMIDI制御が可能となっている。

●MIDI入出力

MUSICDRV.Xは音楽演奏をするだけでなく、MIDI信号の入出力を制御するための基本的な機能をも備えている。

たとえば、楽器にエクスクルーシブメッセージを送って音色を変更したり、楽器からの信号をバルクダンプすることが簡単にできる。さらに、入力されたMIDIの演奏信号に対し、おおまかな時間情報を加えたものを「録音」することも可能(ただし、直接の再生は不可能)。

●FM音源の拡張機能

FM音源に対してもMIDIライクなベンドやモジュレーションが指定できる。

使用方法

CONFIG.SYSに、

DEVICE = MUSICDRV.X #128 #P64

のような指定を加えると、トラックバッファ容量128Kバイト、サンプリングデータ容量64Kバイトのメモリを確保してMUSICDRV.Xが組み込まれる。

または、コマンドラインから、

A>MUSICDRV #128 /P64

と指定しても同様の効果がある。

コマンドラインから組み込んだ場合なら

A>MUSICDRV −R

のようにドライバを解放してメモリを節約することも可能。

OPMDとコンパチで使いたいという場合には組み込み時のオプションを、

MUSICDRV #128 /P400 /Y1 /OPM

のように指定すること。

さて、次にBASICを拡張しなければならない。これには、BASIC.CNFに、

FUNC = MUSIC1

という行をつけ加えるだけでよい。

従来のMUSICDRV.X用のBASIC外部関数はMUSIC2.FNCという名前だった。シャープにはMUSIC2.FNCというファイル名は使わないでほしいといっておいたのだが、C compiler PRO-68K ver.2.0に付属する新しい外部関数もMUSIC2.FNCという名前になっていた。

機能/性能的に見るなら、わざわざMIDI無用時にOPMDRV2.Xを使う人もあまりいないとは思うが、ここは本家の顔を立ててMUSICDRV.X用の外部関数は違う名前に変更する。これ以降、MUSICDRV.X用の外部関数は"MUSIC1.FNC"と呼ぶ。これなら絶対に名前がぶつかることはない。

さて、すでにあるOPMファイルを演奏する場合なら、

A>COPY ABC.OPM OPM1

のように"OPM1"というファイル名にコピーを行うことで演奏が可能だ。

注意点

たとえば、BASICやC言語から音楽演奏をさせていたアプリケーションをコンパイルした場合MUSICDRV.Xではエラーが発生する。OPMDRV.Xを直接呼ぶようなものは実行できない。

使い方にもよるのだが、MUSICDRV.XではOPMDRV.Xよりも多量のメモリを消費する。トラックバッファは大きめにとっておいたほうがよい。

そのほかの詳しい仕様についてはDISK2の¥MUSICDRV¥MUSIC.DOCを参照してほしい。

カードゲーム　KLONDIKE(DISK2)

柴田雅隆

KLONDIKEはソリティア(一人遊び)の一種です。見覚えのあるトランプ遊びですが、やってみると不思議不思議、つい一時の経つのを忘れて没頭してしまうことでしょう。このプログラムは福岡県の柴田雅隆さんによる投稿作品です。

●

DISK2を立ち上げ、KLONDIKE.Xのアイコンをダブルクリックしてください。起動すると、写真1のようにカードが並べられます。7列の場札とその上部に4つの灰色い枠があるのがおわかりでしょう。ゲームの目的は、この枠の位置にすべてのカードを4つのスート(スペード、ハート、ダイヤ、クラブ)ごとにAからKINGの順で積み上げることというもので、これをファウンデーションといいます。写真3が完成の状態です。

ではその手順ですが、場札の各列いちばん上の表向きになったカードに注目してください。写真1の場合、さっそくクラブのAが出ていますので上部の枠のどれかに上げてしまいましょう。以後、クラブの2、3、4……と同じスートのカードが揃えば次々重ねていくことができます。場札からカードが移動すると、その下にあったカードが表向きとなります。

さて、場札のなかでも表を向いたカードは一定のルールでカードを移動でき、ここでは数字が小さくなる順に、しかも赤札と黒札が交互になるように重ねることができます。たとえば写真1の場合ですと、左から1列目のハートの9は3列目のスペードの10の上に重ねることができます。この場合ですとハートの9が移動した跡は空きスペースとなりますが、KINGがあれば、ここに移動することができます。

さらに、重なったカードはまとめて移動することも可能です。たとえば、黒7、赤6、黒5と重なった3枚のカードは、赤の8があればその上に移動できるわけです。

操作は簡単。マウスのカーソルを移動したいカードの上に合わせ、左ボタンを押しながら動かすと目的の位置までカードを移動することができます。重なったカードをまとめて移動するには表を向いたいちばん下のカードにカーソルを合わせ右ボタンでまとめて移動します。

また、画面の右上にはストックがあり、ここをクリックすると、3枚ずつカードがオープンになり、ここからもカードを移動することができます。移動可能なカードがなくなったらストックをクリックしてください。ストックが1周したらNEXTと表示された部分をクリックします。2周目は2枚ずつ、3周目は1枚ずつオープンしていきます。

ストックが3周するとENDマークが表示されます。あとは裏向きのカードを慎重に見比べて可能なかぎり移動を試みてください。移動可能なカードがなくなればゲームオーバーです(写真2)。

なお、いったんファウンデーションの位置に上げたカードでも重なりの条件を満たすかぎり、場札の位置に戻すことも可能です。うまく利用すれば上がる可能性が高くなるでしょう。

注意：このプログラムはCARD2.FNCおよびCARDDRV.Xを使用しています。添付したDISK2を起動すると、これらは自動的に組み込まれますが、他のシステムで使用する場合には各自CARD2.FNCおよびCARDDRV.Xを組み込んでください。



写真1　K画面状態



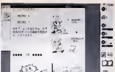
写真2　残念、ゲームオーバー



写真3　めでたく完成



PICファイルの表示例



CUTファイルの表示例

PIC & OPM & CUT

DISK1に含まれるPICファイルはOh!Xのスタッフによる作品だ。VS2.Xとともに掲載されたVSPIC.Xのデモ用サンプルとして収録された。これらはごく当たり前のPICファイルなので、PIC.Xがあれば通常のコマンドラインから表示することもできる。

ForYou.PIC(福岡潤)

付録ディスク用に依頼したもの。

Weeds.PIC(前川潤)

当初はあまりに大きかったので、編集部で階調を落としレタッチしてファイルサイズを縮小したもの。残念ながら原作の色合いや立体感がやや損なわれている。

SLEEPCAT.PIC(山田純二)

山田氏の傑3部作のうちいちばん小さかったもの。

TREE−3D.PIC(丹明彦)

トランスピュータを駆使したレイトレーシング画像。マッピングなどの小技に頼らないところに迫力がある。作成中のマスターディスクを連続セクタに並べ替えたら思いのほか空き領域が出てきたための穴埋めという説もあるが、やはり作品の質が高かったため収録されたもの。

同様にDISK1に含まれるOPMファイルは、岡毛直輝氏のオリジナル曲「イトシキリリモ」と立田正之氏による「小フーガ」だ。著作権関係がほとんど絡まない選曲が特徴的。

また、各ディスクに含まれるカットファイルは、山田純二、福岡潤、高橋哲史の各氏が下絵、取り込み、データ変換、ドット修正まで担当してくれたものだ。

CARD2.FNC&CARDDRV.X

Mouchi Toshiyuki
毛内 俊行

好評のCARD FNCがコンパイラ対応となりました。元々、コンパイルできそうにないプログラムを除き、手軽に高速化が可能です。加えて256色データのサポート、カード枚数の拡張、そしてカードエディタも揃いました。これでほぼあらゆるカードゲームに対応が可能です。

「カードゲームを作るとき、いちいちカードデータを作成するのがうっとうしい」という思いのもとにCARD FNCは生まれました。そしてその思いは、カードゲームを作ろうとするすべての人に共通した思いだということが、CARD FNCを使った数々のゲームプログラムの投稿によって証明されたのではないでしょうか。

そこで、このCARD FNCをもっと本格的に使ってやろうということで、今回早速CARD FNCのバージョンアップ版を発表させていただきます。その名もずばりCARD2 FNCです。

CARD2.FNCの特徴

まずは、CARD2 FNCがCARD FNCと比べてどのように変わったかを見ていきましょう。

1) CARD FNC対応のプログラムがそのまま実行できる

CARD2 FNCはCARD FNCのアップバージョンなのですから、従来のソフトはまったく改造なしに実行することができます。

2) カードデータファイルに対応した

CARD FNCではカードデータを変更するのにはcgetやcsetといった命令を使ってカードを1枚ずつ変更するしか手段はありませんでした。しかし今回、CARD2 FNCに対応させたカードデータファイルの標準形式を設定したことにより、あらかじめメモリにロードしたカードデータを自由に選択することができます。



このカードファイルのおかげで、従来は最大60枚までしか使えなかったカードデータが、今度はメモリの許す限り枚数を気にせずに使えるようになりました。なお、カードデータファイルの作成には、付属のカードエディタを使うと便利です。

3) 従来の16色モードのほかに256色モードにも対応した

CARD FNCは、グラフィックモードが16色表示のときでなければカードの表示ができませんでした。しかしカードデータファイルに、あらかじめ使える色数を設定しておくことにより、従来の16色対応のデータ以外に256色対応のデータも使うことができるようになりました。ただし、65536色モードでのカードの表示は不可能です。

なお、16色対応のカードデータを256色で表示したり、その逆を行ったりすることは可能ですが、表示される絵はめちゃくちゃになります。

4) コンパイラに対応した

コンパイラ用ライブラリの完成により、悲願ともいえるコンパイラへの対応ができるようになりました。もちろん、X BASICからだけではなくCでも使うことができるようになりました。

こんなふうに従来CARD FNCで、不満だと思われていた所を、今回は徹底的に改良しました。しかしCARD FNCでもそうでしたが、カードデータをメモリに置くため、メモリの少ないユーザーの方には使いづらいかもしれません。

特に今回はCARD FNCのときと異なり、カードデータの管理は専用のデバイスドライバが行っています。そのためにX BASICを起動する前に必ずデバイスドライバを組み込んでおかなければならなくなってしまい、少々煩わしくなってしまいました。このドライバはコマンドラインから簡単に組み込むことができます。

とにかく、CARD FNCとメモリの問題というのは切っても切れない関係です。メモリの少ない人（特にメモリ1Mバイトの人）は、OPMDやSX WINDOWなどとの同時使用は諦めて、どうしても使いたい場合はカードドライバを解除してから使ってください。

CARD2.FNCを使う

まず技術的なことはあと回しにして、実際にCARD2 FNCを使うまでの手順を説明しておきましょう。

1) X BASICに関数を組み込む

CARD2 FNCは外部関数ですから、使う前にX BASICに組み込んでおく必要があります。この作業は1回やってしまえばあとは必要ないので、早速やってしまいましょう。

最初にX BASICのディレクトリにCARD2.FNCをコピーします。そしてエディタから、BASIC.CNFなどのコンフィギュレーションファイルを呼び出し

FUNC＝CARD2

の1行を追加します。いままでCARD FNCを使っていた人は、

FUNC＝CARD

の1行を変更してください。これで、次回からX BASICを起動すれば、自動的にCARD2 FNCがX BASICに組み込まれます。

2) ドライバを組み込む

CARD2 FNCにはCARDDRV.Xという、専用のデバイスドライバが必要です。このドライバは、カードファイルをメモリにロードして、CARD2 FNCからの指令に対し、表示するカードのデータなどを返しています。

このドライバが組み込まれていないとき、CARD2 FNCはエラーを出すようになっていますので、CARD2 FNC（またはCARD FNC）対応のプログラムを実行するときには、あらかじめこのドライバを組み込む必要があります。組み込み方は簡単で、コマンドラインから

CARDDRV データファイル名

と、実行すれば簡単に組み込むことができます。たとえば、付録ディスクにTR.DATというCARD FNCのデータとコンパチなトランプデータファイルがあります。このファイルを組み込むときは、

CARDDRV TR.DAT

と入力してやればいいのです。ただし、カードファイルがカレントドライブにない場合は、ファイル名と一緒にパス名まで指定してください。

また、ドライバを解除したい場合は、

CARDDRV /R

というように、ファイル名の代わりにRオプションを付けます。CARD2 FNCを使わないときは、こうしてドライバを解除すると、カードデータがメモリから開放されてメモリに余裕ができます。新しいカードデータをロードするときも、このように一度ドライバを解放しなくてはいけません。

さて、これでドライバの組み込み方がわかりました。ただし、X BASICを起動したり、コンパイルしたプログラムを実行したりするたびにドライバをいちいち起動していたのでは大変ですから、AUTOEXEC.BATなどのバッチファイルに記録させてしまうのが楽でしょう。

ただし、このドライバは特殊な構造をしているために、CONFIG.SYSからDEVICE命令での登録はできないようになっています。組み込みは必ず、バッチファイルかコマンドラインから行うようにしてください。

3) コンパイラを使う

次にコンパイラのライブラリの使い方を説明します。CARDLIB.Aというファイルがそのライブラリで、使い方は、

CC ファイル名　CARDLIB.A

と、コンパイルするときにコンパイルするファイルの後ろに並べて書きます。もし面倒な場合は、BASLIBにアーカイブしてしまいましょう。

さあ、CARD2 FNCの使い方がわかりましたか？　早速ないか遊んでみましょう！

CARD2.FNCの命令

次にCARD2 FNCを使ってプログラムを書く人のために、その命令を紹介していきましょう。

CARD2.FNCでは従来のCARD FNCと比べて、X-BASICで使うことのできる命令の数が増えましたが、もちろん従来の命令もそのままサポートしています。ただし256色対応になったり、ここでは、その従来の命令も含めてCARD2 FNCで使うことのできる5種類の命令を紹介します。

■c_put(x, y, n)

座標（X, Y）で示したグラフィック座標にカード番号nで示したカードパターンを表示します。パラメータはすべてint形式の数値で、返り値はありません。付録のトランプデータを使うときのカード番号と札の関係は、

n＝スート番号×13＋数

で、スート番号はスペード＝0，ハート＝1，ダイヤ＝2，クラブ＝3となっています。

■c_get(n, ca)

カード番号nのカードデータを配列変数caに取り込みます。nはint形式の値、caはchar形式の変数です。返り値はありません。カードデータの大きさは48×96ドットなので、16色モード時で2304バイト、256色モード時で4608バイトが必要です。なお、取り込んだパターンはX BASICのget put命令で扱うことができます。

■c_set(n, ca)

カード番号nに、配列変数caに格納されているデータをセットします。つまりc_get命令と逆の動作をするわけです。その他、パラメータの形式や返り値など、c_get命令と同じです。

さて、以上の命令はCARD FNCでも使われていた命令です。CARD2.FNCではそれ以外に以下の命令が追加されました。

■c_palet( )

表示するカードのパレットを、あらかじめファイルに用意されたパレットテーブルに従って変更する命令です。

ちょっとわかりにくいかもしれませんが、CARD FNCのトランプデータを例に挙げて見てみましょう。あのデータでは黒の表示部分がカラー番号1（暗い灰色）に設定されています。なぜならば、X68000ではカラー番号0が、黒でなく透明色に設定されているためです。実際にカラー番号を0にしてカードを表示したとすると、バックの色が透けて見えてしまい、情けない結果になってしまうはずです。

99をはじめ、いままで発表されたゲームのほとんどは、palet(1-0)などの命令を実行して、カラー番号1を黒くしていますが、色数が最大256色に増えた場合、いちいちパレットを設定するのはやっかいです。あらかじめカードデータファイルには、パレットテーブルを用意してあるので、この命令を実行したらそのテーブルを見てパレットを書き換えるのです。

まだわからない人は、X-BASICを起動して、コマンドラインから、

```
screen 1,1,1,1
c_put(100,0,1)
c_palet( )
```

と実行してみましょう。綺麗い表示されたスペードのAが、きりっと引き締まるのが見えるはずです。

なお、この命令にはパラメータも返り値もありません。

■c_stat(n)

現在使っているカードファイルのステータスを返します。nはint型の数で0か1を代入します。n＝0のとき、返り値はカードの使っている色数で、必ず16か256が返ります。またn＝1のときの返り値はカードの枚数で、返り値は最大カード番号＋1の値が返ってきます。CARD2.FNCでは使えるカードの色数や枚数が固定ではありませんので、この命令を使ってカードのパラメータを確認できるようにしました。

以上がCARD2 FNCで使える命令です。CARD FNCの命令に加えて、さらに強力な命令がサポートされました。皆さんもこれらの命令をどんどん使ってください。

デバイスドライバの仕様

さて、ここからしばらくは少し専門的な話になりますので「僕には関係ないや」と思う人はこの先のカードエディタの解説まで読み飛ばしても結構です。ここではマシン語でカード管理ドライバを操作したいという人のために、ドライバの使い方などを説明していこうと思います。

1) デバイスドライバのコマンド

カード管理ドライバCARDDRV.Xのコマンドコードは2バイトの数字から成ります。データをやり取りするときは、読み込み/書き込みモードでドライバをオープンし、2バイトのコマンドをドライバに書き込んだ直後に4バイトのデータをデバイスから読み込みます。ちなみにCARDDRV.Xのデバイス名は“CARDS”です。

また、コマンドの中にはコマンドコードのほかに2バイトのパラメータを一緒にドライバに転送するものがあります。このときは、書き込みバイト数を4バイトとして、コマンドコード－パラメータの順にドライバに書き込んでやります。それでは各コマンドの説明をしていきましょう。

■コマンドコード$0000
カードデータアドレスの読み込み

指定したカードのパターンが格納されて

いるメモリアドレスを返します。パラメータとしてコマンドコードと一緒にカード番号を2バイト渡します。もし、存在しないカード番号を渡した場合は、デバイスから－1が返されます。

返されたアドレスからのデータは、IOCSコールの_PUTGRMを使って表示することができます。

●コマンドコード$0001
パレットテーブルアドレスの読み込み

カードのパレットを定義するパレットテーブルの先頭アドレスを返します。パラメータは存在しません。

●コマンドコード$0002
カードファイル情報の読み込み

カードファイルの情報を読み込みます。返ってきたデータの最初の2バイトはカードの色数、続く2バイトがカードの枚数を示しています。

●コマンドコード$FF00
ドライバを解除するための情報の読み込み

このコマンドはすでにメモリ上にあるCARDDRV.Xを解除するために必要な情報を読み込むためのもので、普段は使いません。このコマンドを実行したときに限り特別に、ドライバからは12バイトの情報が返ってきます。

情報は最初から4バイトごとに、カードデータの先頭アドレス、CARDDRV.X本体の先頭アドレス、直前のデバイスドライバのリンクポインタのアドレスを示しています。CARDDRV.Xを解除するときはこの先頭2つのアドレスをスタックにセットして_MALLOCを実行し、最後の4バイトが示すアドレスに、CARDDRV.Xのリンクポインタの中身をコピーしてあげればいいのです。

以上がドライバのコマンドです。マシン語でカードを表示したい人は使ってみてください。

2) カードデータファイルの仕様

ドライバが管理するカードデータファイルはそのフォーマットが決まっています。もし、自分で独自にカードデータファイルを作成したりするときは、ファイルのフォーマットをこれから説明する仕様に合わせてください。

カードデータファイルは、ヘッダ、パレットテーブル、カードデータの順に、3つに分けることができます。これからそれらを順に説明していきましょう。

●ヘッダ

ファイルの先頭から16バイトがファイルのヘッダ領域です。ファイルヘッダは先頭から順番に以下のように分けられます。

0～3バイト	文字列"CARD"
4～5バイト	カードデータの色数
6～7バイト	カードデータの枚数
8～15バイト	未使用領域

最初の4バイトの文字列は、このファイルがCARDDRV.X用のカードデータであることを示すもので、先頭の4バイトは必ず大文字で"CARD"と書かれていなくてはいけません。それからはワード単位で色数、枚数という順に格納されており、それ以降は将来の拡張を考えて未使用ということになっています。

●パレットテーブル

ファイルヘッダの後ろに続くのがパレットテーブルの領域です。ちなみに、CARDDRV.Xはカードデータファイルをメモリ上にロードするとき、このパレットテーブル以降を、そっくりそのままロードします。

パレットテーブルには、各カラーコードのパレットデータが入っており、c.paletなどの命令でパレットを設定するときに使われます。パレットコードは$0000～$FFFFのワード単位で表記されますから256個のパレットを表すためには、テーブルの大きさは全部で256ワード(512バイト)必要になります。

パレットテーブルは先頭からカラーコード0、1、…255の順に並んでいます。カードデータが16色のときはカラーコードが16個しか必要ありませんから、テーブルの先頭から32バイト分のパレットデータがセットされ、それ以降には0が入ります。

●カードデータ

パレットテーブルの後ろはカードデータ領域です。データ形式はX-BASICのget/put命令やIOCSコールの_PUTGRMで使うものと同じ形式（16色のときは1バイト＝2ドット、256色のときは1バイト＝1ドットのカラーコードを横方向に並列したデータ）ですのでカードデータは1枚につき、2304バイト（16色）、4608バイト（256色）の大きさが必要となります。

* * *

以上がカードデータファイルの内容です。カードエディタを使えば、こんなことは考えなくてもいいのですが、たとえば既存の絵から、カードデータを作成したりしたいときなどは、このファイルフォーマットを参考にして、自分で工夫をしてみてください。ファイルの構造は簡単なので、X-BASICでも手軽に、ファイルの書き換えプログラムが作れることと思います。

カードエディタを使う

さて、今回CARD2.FNCについた特別付録がこのカードエディタです。付属のディスクにはCDEF.Xというファイル名で入っています（シーデフ・エックスと読んでくれれば嬉しいのですが）。

カードエディタは、コマンドラインからCDEFと入力すれば簡単に起動することができます。エディタを実行すると最初に、エディットするカードファイルの設定を行います。

まず、エディットするファイルが新規ファイルかどうか聞いてきますので、新規ファイルならYES、そうでないならNOの位置にマウスカーソルを合わせてマウスの左ボタンをクリックします。このエディタは、カードファイルのファイル名を入力するとき以外、すべての操作はマウスで行うように作られています。

エディットファイルが新規でない場合は、このあとファイル名を入力してすぐにエディットモードになりますので、ここでは詳しい説明はいらないでしょう。さて、新規ファイルを選んだ場合、エディットモードに入る前に新しいカードデータファイルをディスクに作成しなくてはいけませんので、まず新規ファイルを作成する手続きから説明しましょう。

まず、エディットするファイルのファイル名をキーボードから入力します。するとメッセージウィンドウに

色数　256
枚数　100

と表示されます。このままOKをクリックすると、色数256色、枚数100枚のカードを扱うカードデータファイルが作成されてしまいますので、用途に応じてパラメータを変えてください。変えたい数字の上にマウスカーソルを持ってきてボタンをクリックすれば、その数字を任意の値に変えることができます。

まず色数を変えたい場合、数字の上でマウスボタンをクリックするたびに、256→16→256の順に数が切り替わります。枚数は各桁ごとにボタンをクリックします。

このとき、左ボタンをクリックすると0→1→2……9の順番で数字が切り替わり、右ボタンをクリックすると、9→8→7……0の順番で数字が切り替わります。ただし、100の桁では、数字は0と1が切り替わるだけです。

つまり、このカードエディタで扱うことのできるカードの枚数は、最大199枚ということになります。本当はドライバで扱える最大までカードの枚数をサポートしようと思ったのですが（65536枚）、メモリの容量を考えると、とてもそんなにたくさんのカードは扱えないのでやめました。それでも256色199枚のカードデータファイルを作成するとそのサイズは、896Kバイトと超キングサイズになってしまいます。フロッピーディスクに納めるにはこれくらいが限界だと思います。

さて、ファイルの設定が終了すると、いよいよカードのエディットに入ります。エディットの方法は、各種グラフィックツールやパターンエディタと基本的には同じなのですぐ使えると思いますが、ひととおりの機能の説明を一応しておきます。

1) 色の設定

まず、絵を描くときのペンの色指定の方法ですが、画面の右上にパレットテーブルがありますので、好みの色のところにマウスカーソルを持っていき、マウスの左ボタンをクリックします。

2) 絵を描く

画面の左にある格子画像がエディット画面です。マウスカーソルをエディット画面へ持っていき、左クリックをするとエディット画面上に自由に絵を描くことができます。なお、エディット画面上でマウスの右ボタンをクリックすると、マウスカーソルが指している画面上の色をペンの色として設定できます（スポイト機能）。

3) 画面の編集機能の説明

CDEF.Xには以上の機能のほかに画面の切り替わりやスクロールなど、カードデータの編集用の便利な機能が用意されています。これらの命令は、画面中央にあるメニューを選択することによって使うことができます。以下、それらの命令の説明をします。

なお、これらの機能を使用中、座標などの決定はマウスの左ボタンのクリックで行い、キャンセルやEXITしたい場合はマウスの右ボタンをクリックするか、画面右下に開いている場合はそのウィンドウ内の"キャンセル"や"EXIT"のところで左ボタンをクリックします。

●FILL

指定した長方形の中を指定した色で塗り潰します。この機能を選択すると、マウスカーソルはエディット画面の中しか移動できなくなりますから、画像上で、塗り潰したい長方形のかどを、左上－右下の順で指定してください。

●スクロール

エディット画像を指定方向に1ドットずつスクロールさせます。機能メニューからスクロールを選択すると、画面右下にメッセージウィンドウが開き、上下左右4方向を向いた矢印が表示されるので、マウスカーソルをスクロールさせたい方向の矢印の所へ運んでクリックしてください。

●コピー/ペースト

エディット画面上で指定した長方形の中に描かれたパターンを写し取り、別の場所に貼り付けます。なお貼り付けるときに、写し取ったパターンを上下、左右に反転することができます。

まず、パターンを写し取るには機能メニューのコピーを選択します。写し取る領域の指定は、FILLのときと同じ要領で、左上－右下の順に長方形の指定を行ってください。指定された長方形の内容が、エディタのカットバッファにセットされます。

写し取ったパターンの貼り付けはペーストを使います。機能メニューからペーストを選択すると、画面の右下にメッセージウィンドウが開き、写し取ったパターンが表示され、その横に"↑↓"と"←→"の2つの矢印が表示されます。"↑↓"の上にマウスカーソルを持っていき左クリックをすると、ウィンドウ内のパターンが上下反転を、"←→"でクリックをすると、左右反転をするのが見えます。

貼り付けるときはマウスカーソルをエディット画面の中に持っていき、任意の場所で左クリックをします。ただしマウスカーソルがエディット画面の中にあっても、貼り付けるパターン全体がエディット画面に入りきらないときは、クリックをしても貼り付けは行われません。貼り付けが可能なときは、エディット画面上に貼り付ける領域の枠を赤く表示します。

またパターンの貼り付けは、連続して何カ所にも貼り付けることができるように、1回貼り付けてもエディットモードには戻らないようになっています。エディットモードに戻りたいときは、右クリックをするか、メッセージウィンドウの"EXIT"のところで左ボタンをクリックしてください。

●パターンのGET/SET

すでにカードデータファイルにあるカードパターンをエディット画面上に取り出したり、エディット画面上で描いたカードパターンをカードデータファイルにセットしたりします。

カードデータファイルからエディット画面上にパターンを取り出すにはGETを使います。選択すると、画面いっぱいにファイルの中に格納されているカードが表示されるので、任意のカードの上にマウスカーソルを持っていき左クリックをします。

ただし1画面に表示できるカードの枚数は40枚までです。カードの枚数がそれ以上のときで、41枚目以降のカードを選択したいときは、画面の右下の"NEXT"をクリックしてください。

エディットしたカードパターンをファイルにセットしたいときはSETを使います。まず、GETのときと同じように画面いっぱいにカードが表示されますので、任意のカードデータのところへマウスカーソルを持っていき左クリックをすると、エディットしたカードがファイルにセットされます。

ただしプログラムの都合上、256色モードでデータのセットを行うと、ファイルをセットするのに5～6秒の時間がかかります。16色のときは気にするほど時間はかかりません。

4) エディットするファイルの変更

パレットテーブルの下に、現在エディット中のファイルの名前が表示されていますが、そのファイル名の上でマウスをクリックするとファイルの変更が可能です。設定方法はこのエディタを起動したときとまったく同じで、新規ファイルかどうかの確認やファイル名の入力などを行います。

5) 終了

エディットが終わったら、エディタを終了しHuman68kへ戻らなくてはいけません。終了させる方法は、機能メニューの下にある"QUIT"を選択します。また、ファイル名の入力時にファイル名の代わりにCTRL-C(BREAK)を入力しても、エディタを終了します。

以上がカードエディタの使い方です。私がCARD.FNCのデータを作る際も、これとほとんど同じ仕様のカードエディタを独自に作って使っていたので、皆さんも根気さえあれば自分独自のオリジナルカードが作れると思います。どんどん活用してください。

プログラムリストについて

参考までに、ドライバのソースリストを載せておきます。が、綺麗なプログラムではないので、あまり参考にならないと思います。特にカードエディタに至っては、締め切りに追われながら無計画に作ったために、プログラムそのものはえらく汚いものになってしまいました。

それでも，このエディタの場合，過去に作ったMZ-2500のプリンタ用外字エディタ（Oh!X1988年11月号）やCARD.FNCのカードパターンを作ったカードエディタなどの経験が生きていたので，プログラムは難くても結構まともなでき栄えになりました。

しかし，いままでまったく手をつけたことのなかったデバイスドライバなどは，参考になるようなプログラムに仕上がったのかどうか。完成したいまでもあまり自信がありません。しかも，都合によりCONFIG.SYSからの登録をできなくするために，デバイスヘッダをデバイスの先頭に持ってきていませんので，俄いプログラムが余計に複雑になってしまった恐があります。

ただ，デバイスドライバなどのソースには，1行ずつちゃんとコメントを入れていますので（これは実は自分のためだったのですが）これらのコメントと命令を見比べれば，見づらいリストも少しは見やすくなると思います。

なお，これらのプログラムの作成にあたり，カードエディタの作成にはX-BASICとGCCのVer.1.36.01を使いました。また，その他のプログラムはすべて，XCに付属のアセンブラ，AS.Xを使っています。

最後に

いやー，今回の仕事は実際とてもヘビーでした。だいたい最初に締め切りまでの猶予期間が1カ月はあったはずなのに，半月たってもドライバが完成していなかったり（いったん完成はしたのだけど，ボツになったのだった）コンパイラのライブラリに原因不明のバグが発生したりと，アクシデントだらけ。最初のうちは「1カ月でできるね？」という編集部の問いかけに，「大丈夫です」とか「まかせてください」などと気前のいい返事をしていたのが，だんだん「頑張ります」とか「やってみます」などという返事が多くなってきて，ついには「本当に完成するのだろうか――」と途方に暮れてしまっていました。

とにかくシステムプログラムというのは作っていて思いっ切りストレスが溜まります。私はこの期間，ストレスが溜まるとYet Another Columnをやりストレスを解消していました。おかしなもので，ストレスが溜まっているときにこのゲームをやるとスコアが飛躍的に伸びます。この期間に記録したスコアはトップの43480点を最高に，4万3千点台を立て続けに3つも記録しました。いま，挑戦してもこのスコアの足元にも及ばないのですから不思議なものです。

話が横にそれましたが，そんなこんなの苦労のうちに，無事にCARD2.FNCも完成し，こうして無事皆さんに配布することができました。これもすべて，私が行き詰まるたびに数々のアドバイスや励ましの言葉をいただいた編集部やそのスタッフの方々のおかげです。ここに改めてお礼申し上げます。特に，本誌連載のX68000マシン語プログラミングは，デバイスドライバを作ったことのない私にとってプログラマーズマニュアルと並ぶ数少ない資料として活用しました。村田敏幸氏に深く感謝いたします。

その他，開発期間中に数々のわがままを聞いてくださった人たち，特に日本大学コントラクトブリッジクラブの方々に深く感謝します。それでは皆さん，これからもCARD2.FNCをよろしくお願いします。

リスト1

Z'sSTAFF支援ツール"Z's-EX"

グラフィックツールの最高峰としてそびえるZ'sSTAFF PRO-68K。でも、"PRO"ならばもっと強力な機能がほしいもの。そこで日頃もの足りなく思っていた機能のいくつかを拡張/強化してみましょう。残念ながら、完全に使うには4Mバイト（?）のメモリが必要です。

Tan Akihiko
丹 明彦

第1章　今回のプログラムは

今回のプログラム「Z's EX」は、Z'sSTAFF PRO-68Kの機能を拡張し、ユーザーが絵を描く作業をサポートするツールです。

拡張といっても、Z'sSTAFFそのものを改造するわけではありません。Z's EXは、通常はメモリ空間のどこかで待機しています。そのあいだは、ふつうとまったく同じように描画作業を行うことができます。ユーザーは、Sキーを押すことでZ's EXを呼び出すことができます。Z's EXを呼び出しているあいだは、通常のZ'sSTAFFの機能を使うことはできません。Z's EXでの作業を終えて、Z's EXを抜けて初めてZ'sSTAFF本来の作業を続けることができます。

Z's EXはZ'sSTAFFとはまったく独立に動作してユーザーの作業を助けるアプリケーションです。

Z's-EXは、呼び出し元のプログラムがZ'sSTAFFかどうかを一切チェックしていないので、Z's-EXはZ'sSTAFFのバージョンに関係なく使用できるはずです。原理的にはZ'sSTAFFでなくてもいいのですが、それはちょっと事情があってできません。

このプログラムのアイデア、つまりZ'sSTAFFと同居して独立に動作するアプリケーションという考え方は、電脳倶楽部に掲載された「PICFILER」というプログラムから拝借しました（実はソースリストからも少々失敬しています）。これは、いまやX68000の標準画像フォーマットとして定着した観のあるPICファイルを、Z'sSTAFFからロードおよびセーブできるようにしたツールです。これを使うことで作業効率は飛躍的に上がりました。正直いって、Z'sSTAFF標準のZIMファイルは作業をするうえでネックになっていたのですから。

Z's EXは、このPICFILERを下敷きにして、あると便利そうな機能をいくつか付け加えたものです。編集部で検討の結果、今回のバージョンでは以下の機能を持たせることにしました。

・PICFILER——これは当然です。

・裏画面とその切り替え——Z's-EXは、ちょっとした作業場として、また後述する画面合成のソースとして、裏画面を1枚持っています。したがって、システムはZ'sSTAFFで使うメイン画面（G-RAM）と裏画面の2枚を持つことになります（厳密にいえばまだほかにも画面はあるのですが、それはひとまず置いておきます）。両者はいつでも切り替えられます。切り替えれば、裏は表に、表は裏になるのですから、当然、どちらの画面に対してもZ'sSTAFFの機能は使えることになります。

・3次元マッピング——裏画面を1枚の長方形と考えて、画面の中に広がる空間に伸ばしたと思ってください。その板は回転も移動もできます。個所が決まったら、そこに張り付けることができます。すると、裏画面の絵がパースのかかった状態で出現するものです。

ちなみにこれは、Z'sSTAFFの変形とは違います。あちらは文字どおりの変形で、用途がまったく別なのです。

なお、移動も回転もしなければ、それは裏画面からのコピーと同じになります。マスキングを上手に使えば、画面合成にも応用できます。なお、コピー用のときだけは高速に処理されます。

・マスキングのペイント　Z'sSTAFFはマスキング機能を持っています。あれはなかなか強力なのですが、どうも使い勝手が悪い。たとえば人物の絵を画面の真ん中に描きます。その人物の周りだけをマスクしないと思ったとします。当然ペイントしたいと思います。できません。いやできないというのは正しくありません。正確には全画面がべったりとマスクで塗りつぶされてしまいます。しかたがないので、ペンを使って人物の輪郭に沿って丁寧にマスクし、しかるのちに周りをペイントします。ああ文化的。

マスクは作業画面とは独立に描画するという考えで作られています。これはこれで間違ってはいません。しかし不便に感じる局面が多く発生するのも事実です。そこで、単色領域を塗りつぶしてマスクに変えるペイントルーチンを用意しました。マウスでちょいと指定するだけで一瞬にして人物の周りだけをマスキングできるのです。これは画面合成の際、大いに威力を発揮することでしょう。

・特殊効果——あったら面白そうな画像処理機能を4つほどつけてみました。いずれも、マウスで囲んだ領域の内側に処理を施すようになっています。

1）白黒化処理　読んで字のとおりです。

2）ランダムフラクタル処理　本誌1990年9月号の特集で紹介したものを少し高速にしたものです。自然物の疑似的な表現や、レイトレーシングのためのマッピングデータ作成ツールとしてお使いいただけることでしょう。

3）フレア処理　明るい部分を光源に見立て、その周りにぼんやりと光を滲ませる処理です。光源部（?）はマスクで指定します。

4）微分処理　名前は難しそうです。しかしできあがりは単純明快。石でできたレリーフのような効果が出せます。取り込み画像に対してかけると、なかなか味のある絵になります。


メインメ

第2章　インストールのしかた

さて、Z's EXを使うためには、お使いのシステムにインストールしなくてはなりません。

インストール作業は簡単です。Z's EX.Xとrf.datおよびmapcon.datをZ'sSTAFFと同じディレクトリに置くだけです。ただ、ディスク容量の関係からフロッピーで使う場合にはそれなりの工夫が必要なこと、そして、rf.datは付属の自動生成プログラム（rfbuild.x）を使って作らなくてはならないこと、この2点には注意しておいてください。

（メインメモリ1Mバイトの方へ）

Z's EXは、当然のことながらZ'sSTAFF単独で使っているときよりもメモリを消費します。そこで非常に残念なお知らせがあります。Z's EXシステムは、メインメモリ1MバイトのX68000では動作しません。

もうこれはどうしようもありません。裏画面に512Kバイトも取っているのですから、まあ当然の結果とはいえますが。

（メインメモリ2Mバイトの方へ）

メインメモリ2MバイトあればZ's-EXシステムは動作します。ただ、依然として厳しい状況には変わりありません。アンドゥも使えないようです。裏画面を有効に使って対処してください。

（フロッピーディスクでお使いの方へ）

Z's EX.Xは起動時に2つのデータファイルを読み込みます。rf.datとmapcon.datです。そのあと、STAFF68K.X（Z'sSTAFFの実行ファイル）をチャイルドプロセスとして呼び出します。これらのファイルがないとエラーになってしまいます。

以上の理由から、ソースリストを書き換えて再コンパイルしない限り、STAFF68K.XはZ's EX.Xと同じディレクトリに置いておかなくてはなりません。また、rf.datとmapcon.datも同じディレクトリに置いておく必要があります。

Z'sSTAFFのシステムディスクの容量は、ただでさえかなり厳しい状況にあります。そこへ持ってきて、Z's EXのけっこう大きなファイルをいくつも転がさなくてはなりません。したがって、これまたまことに残念な話ですが、Z's EXとZ'sSTAFFが両方入ったシステムは、1枚のフロッピーディスクには入りません。

ハードディスクユーザーの人には、まったく問題はないのですが、フロッピーディスクだと、まずインストールから悩まなくてはなりません。

（というわけで解決編です）

現在のX68000ユーザーの多くは、「メインメモリ2Mバイト、ハードディスクなし」というシステム構成を取っていると思います。したがって、上で述べたようなトラブルに巻き込まれる可能性は非常に大きいものといわなくてはなりません。

この問題は、ディスクを2枚組にし、なおかつ最低限のデバイスドライバを組み込み、使える記憶領域を最大限にすることで解決することにします。

以下の説明は、「メインメモリ2Mバイト、ハードディスクなし」のシステムに対するインストール方法についてです。メモリがもっとあるとか、ハードディスクをつけているとか、そういう人は適宜変更してください。

2枚組になったディスクのうち、AドライブにはHuman68kのシステムやASK68Kの辞書、BドライブにはZ'sSTAFFおよびZ's EXの実行ファイルを入れることにします。図1をご覧ください。両ドライブに入れるべきファイルの一覧と、CON

2MバイトRAM、ハードディスクなしのユーザーのための
システムディスク構築例

ディスクは2枚用意します。
Z's EX 関係のファイル以外のファイルは、Human68k システムディスクおよび Z'sSTAFF PRO-68K システムディスクからコピーしてください。

```
【ドライブ0のファイル】
¥
├─HUMAN.SYS                以下は Human68k のファイルです。
├─CONFIG.SYS
├─COMMAND.X
├─AUTOEXEC.BAT
├─SYS
│  ├─FLOAT2＋.X            ＊もちろんFLOAT2.XやFLOAT3.Xでも構いません。
│  └─ASK68K.SYS
└─DIC
   ├─X68K_M.DIC
   └─X68K_S.DIC
【ドライブ1のファイル】
¥
├─STAFF68K.X               以下は Z'sSTAFF PRO-68K のファイルです。
├─STAFF68K.SYS
├─CHRM.DAT
├─ICON.DAT
├─ICONG.DAT
├─ICONB.DAT
├─ICONR.DAT
├─MPALET.PAL
├─MPALET.WPL
├─PCSHOW.PDT
├─PEN.PN3
├─TILE.TIL
├─TONE.TON
├─STAFF.TMP
│                          以下は Z's-EXのファイルです。
├─Z's_EX.X                ← 付録ディスクに入っています
├─mapcon.dat               ← 付録ディスクに入っています。
└─rf.dat                   ← 付録ディスクには入っていません。
                             rfbuild.x を用いて作ってください。
【A:¥CONFIG.SYSの内容】
FILES      =20
BUFFERS    =20 1024
＊DEVICE   =A:¥SYS¥ASK68K.SYS A:¥DIC¥X68K_M.DIC A:¥DIC¥X68K_S.DIC
DEVICE     =A:¥SYS¥FLOAT2＋.X
【A:¥AUTOEXEC.BATの内容】
ECHO OFF
B:
Z's_EX
A:
```



FIG SYSおよびAUTOEXEC BATの内容が書いてあります。これにしたがってディスクを作り上げてください。メモリを食うようなものはできるだけCONFIG SYSに書かないよう心掛けてください。それだけZ'sSTAFFで使えるメモリが減りますので、いろいろと制限が出てくるのです。実のところ ASKもあまりすすめられないのです。もちろん、4Mバイトほどメモリを積んでいれば、なんの支障もありません。

この章の最初で申し上げたとおり ランダムフラクタル用のデータファイル（rf dat）は自分で作らなくてはなりません。付録ディスクを捜索しても、ついてこないのです。実はrf datは なぜかZ's EXシステムのなかではもっとも大きなファイルになってしまいました。しかしその正体は、乱数で作った配列にすぎません。ですから容量の限られた付録ディスクに入れるわけにはいきません。代わりに、rf datを自動生成するプログラム、rfbuild xを入れてあります。適当なドライブにrfbuild xをコピーしておいて

rfbuild

としてください。なにかメッセージが出てきますが 無視してください（実は オプション指定で ランダムフラクタルの形状を変えられるのです。しかし基本的に必要なさそうなので オプションなしで実行してください）。そのうちにrf datができてきます。それを、他ドライブのZ's-EXと同じところにコピーしてください。以後はrfbuild xは不要です。

それ以上の説明は不要でしょう

第3章 使い方

（システムの起動）

Z'sSTAFFとZ's EXのあるディレクトリ（前述のとおり 同じディレクトリにファイルを置いておいてください）に行き そこでコマンドラインから

Z's EX

とします。あとは自動的にZ'sSTAFFが立ち上がります。先ほどのインストール手順どおりにされた方は AUTOEXEC BATにここまでは書いてありますので、スイッチを入れてしばらくすると立ち上がるはずです。もし、グラフィックRAMを保存して立ち上げたいのであれば、この時点でスペースキーを押しておいてください。

起動してしまえば、基本的にディスクドライブは自由です。PICファイルの入ったディスクに入れ換えても大丈夫です。ただ文字入力をする際は辞書ファイルが必要なので そのときにはAドライブを戻さなくてはならないでしょう。

また Z'sSTAFFは、線長の線を編集するときに、巨大なテンポラリファイルを作ります。これはいまはBドライブに作っているので どうしても削除にしたいのなら Bドライブも戻す必要があるでしょう。むろん これらの機能を使わないのなら、起動してすぐにディスクを抜いてしまっても害はないと思われます。メモリが少ない場合はなおさらです。余分なものを組み込むべきでもありません。

起動した状態では、グラフィックもマウスもZ'sSTAFFの管理下にあります。お好きなように絵を描いてください。

（メインメニューの起動と終了）

Z's EXシステムはZ'sSTAFFから呼び出しますが その際ひとつだけ注意することがあります。

Z's EXシステムを呼び出す前には 必ずZ'sSTAFFのすべてのウィンドウを閉じてください。ウィンドウ左上のタイトル部分をダブルクリックすれば閉じます（Z'sSTAFFを使い慣れた人には こんな説明は不要ですが）。メニューバー（起動時 画面の上端に出る長いもの）も消します。これにはスペースキーを押します。

これは厳守してください。それがもとで暴走するという性質のものではないのですが、画面がめちゃくちゃになってしまう可能性があります。なぜなら Z'sSTAFFのウィンドウは 一部を除いてG-RAMに描いているからです。

Z'sSTAFFのウィンドウはすべて消す。これは絶対に忘れないようにしてください。

画面上からすべてのウィンドウが消えたのを確認したら そこで初めてSキーを押してZ's-EXを呼び出してください。ビジュアルシェルのウィンドウに似た、Z's EXのメインメニューウィンドウが現れるはずです。

ウィンドウ操作は ビジュアルシェルとほぼ同様にしてあります。ウィンドウを動かしたいときは タイトルバーをつかんでドラッグしてください。ドラッグはおわかりですね？ 再びZ'sSTAFFに戻りたいときはメニュー左上のクローズアイコンをクリックしてください。メニューウィンドウが消えます。もうすでにここでZ'sSTAFFに戻っています。マウスをクリックすればZ'sSTAFFの動作をしますので気をつけてください。スペースキーを押せば、Z'sSTAFFのメニューバーが現れます。

メインメニューにはいくつか単語が並んでいます。これらはZ's-EXで使えるコマンドを表しています。そのなかから 使いたいコマンドを選んでクリックしてください。これから 各コマンドの説明に入ります。

（PICFILER）

ウィンドウの中央に カレントディレクトリのファイルが表示されます。ただし表示されるのはディレクトリとPICファイルだけです。ほかのファイルにはちょっかいを出せません。

ディレクトリを指定するときは、そのディレクトリの名前の左にあるボタン（ただの小さな四角ですが）をクリックしてください。そのディレクトリに移ります。ファイルを指定するときも同様の操作をします。つまりロードまたはセーブしたいファイルの名前の左にあるボタンをクリックします。すると そのファイルの名前がウィンドウ下のファイル名表示エリアに現れます。

このファイルをロードするとしましょう。それには、ウィンドウの[LOAD]アイコンをクリックしてください。ロードするかどうか尋ねてくる（ちょっとうっとうしいのですが 誤操作防止のためです）ので [OK]または[CANCEL]をクリックしてください。

もしセーブしたいなら、ウィンドウの[SAVE]アイコンをクリックすれば、ロードの場合と同様にしてできます。ただ セーブの場合は、当然 新しくファイルを作ることが必要になることが多いのです。そのため ファイル名の指定をキーボードからすることもできます。その場合は ウィンドウ下方のファイル名表示エリアをクリックしてください。ファイル名表示エリアにカーソルが点滅しますので、キーボードでファイル名を打ち込んでください。このとき 拡張子".PIC"はつけてもつけなくてもかまいません。たとえ間違った拡張子をつけても 強制的に".PIC"に変えます。



PICFILER

手抜きなことに、カーソルが点滅しているモードのあいだはマウス操作は効きません。間違えてファイル名指定モードに入ってしまったら、リターンキーを押して抜けてください。

ファイルウィンドウは8行しかありません。でもファイルやディレクトリが8つですむわけはありません。そこで ファイルウィンドウをスクロールさせなくてはなりません。でも ウィンドウのどこを見てもスクロールアイコンはありません。ここでは 電脳倶楽部の標準シェルであるDSHELLで採用されて成功を収め はたまた今回の付録ディスクのVS2（ビジュアルシェル拡張版）のドキュメントウィンドウでも採用されている操作方法を採用しました。

つまり、スクロールアップおよびスクロールダウンをするには ファイルウィンドウ内で左および右クリックするのです。ただし ファイルウィンドウ左端のボタンのところでクリックすると すでに説明したとおり そのファイルが選択されるだけですので気をつけてください（別に害はありませんが）。

ついでに ファイルウィンドウの中で左右のボタンを同時にクリックしてみてください。今回は左右クリックに 親ディレクトリに戻る"CD .."の機能を持たせてみました。もちろん ファイルウィンドウの"¥"ディレクトリを指定しても同じです。

ただ これには異論のある方もいらっしゃることでしょう。DSHELLにしてもVS2にしても 左右同時クリックは「そこから抜ける」という意味です。その論法でいけば PICFILERそのものを抜けてメインメニューに戻るというのが筋でしょう。ただDSHELLでは 左右同時クリックは「前の階層に抜ける」と解釈できなくもありません。それなら、理解化ディレクトリの前の階層に抜けるというこのPICFILERの作法もあながち間違いとはいいきれないかもしれません。DSHELLは ルート（つまり目次ですね）からは左右クリックで抜けられないようになっています。これは当然誤操作でDSHELLを終わってしまわないようにするためのものでしょう。このへんも、ルートディレクトリからは抜けられないPICFILERと共通しているところですね。どうしても我慢できないという方はご意見お待ちしております。もし次のバージョンを制作する機会があれば参考にさせていただきたいと思います。



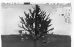
さて話が逸れてしまいましたが、お話し忘れていたことがひとつあります。ドライブを変える方法です。ウィンドウ左上に "A "という表示のある部分 それがそうです。ドライブを変更するには トライブ表示部で左または右クリックしてください。フロッピーディスクの入っていないドライブはスキップします。のはずなのですが、ときどきシステムのほうから「ディスクが入っていません」というエラーを出されてあわてることがあります。Aドライブがフロッピーディスクで、なおかつディスクが入っていないときに出るようです。白状すると プログラムを作るときはAドライブがハードディスクだったので チェックを入れ忘れたのです。

左と右のボタンにまったく逆の機能を割り当てるという作法が 本ツールにおいては今後も頻繁に出現します。左でUP、右でDOWN…… まるでポピュラスですな。

メインメニューに戻りたいときは ウィンドウ左上のクローズアイコンをクリックしてください。PICFILERのウィンドウが閉じ メインメニューが現れます。

（ALTERNATE SCREEN）





裏画面と作業画面を切り替えます。説明の必要もないでしょう。

（MAPPING）

MAPPINGを呼び出すと 上数個のアイコンがつまったウィンドウが開き 画面には妙な枠が出てくることでしょう。試しに上半分に並んでいる 中に図の描いてある四角（アイコン）をいろいろとクリックしてみてください 枠がぐるぐると回ったりずるずると動いたりすることでしょう。左ボタンだけでなく 右でもクリックしてみてください。

左半分の 棒の周りに矢印が回っているようなアイコンは 「座標軸まわりの回転」を 右半分の、矢印が描いてあるアイコンは「座標軸方向の移動」を、それぞれ表しています

座標軸は 上からx、y zの順番になっています（これは そう意識する必要はありません。枠はアイコンに描いてあるとおりに動くようにしているつもりです）。

左クリックは表示してあるとおりの方向に 右クリックは表示してあるのとは逆の方向に移動および回転を行います。これまた右い機能を持たせてしまったのは ある意味では迷惑なのかもしれません。が この下の作業では往々にして「あら動かしすぎたよ」というミスが発生しがちです。そんなときにマウスをもう一方のアイコンのところに持っていって左クリックでもよいのですが、それはなかなか面倒。それよりも空いているボタンで逆の動作をすぐにできるようにしたほうがありがたいのではないかと思ったわけです。

さて、ぐるぐる動かしているうちに、なにがどうなってしまったのかわからなくなることがあります。枠にはひとつ印をつけてあるので どんなになっても方向はわかりますが、また、マッピングする画像を新しいものにかえた場合など いつまでも前の結果を残しておくのは気持ち悪い。で そんなときは [RESET]アイコンをクリックしてください。初期状態に戻ります。枠



光る部分をマスクで指定



処理する　る

は価値一杯　移動も回転も1回もしていない状態に戻ります。新しい気持ちで動かし直してください。

いよいよ絵を張り付けることにします。裏画面の準備はいいですか？　裏画面にマッピングするデータを用意しておかないとなにもなりません（というよりも　裏画面に真っ黒な絵が張り付くだけになります）。準備ができたら　OKアイコンをクリックしてください。

実行は　はっきりいって速くありません。しかし　起動してから初めてMAPPINGに来たとき　それから回転も移動も行っていないとき　移動　回転をRESETしたとき　それから回転も移動も行っていないとき　つまり枠が初期状態にあるときに張り付けると　特別に高速で処理してくれます。これは使えます。画面合成が数秒で行えます。

メインメニューに戻りたいときは、PIC FILERと同様に、ウィンドウ左上のクローズアイコンをクリックしてください。

〈MASK PAINT〉

これを選択すると、メインメニューは閉じます。マスクをかけたいところにマウスカーソルを持っていって　左クリックしてください。その部分にマスクがかかって、青く点滅しはじめます。メインメニューに戻るには　右クリックします。

なお、マスクペイントをかけてからZ'sSTAFFに戻ると　ときどきマスクがうまくかかっていないように見えるときがあります。ウィンドウを動かしたあとに　塗ったはずのマスクがなくなっているときがあるのです。原因はよくわかりません。こんなときは　あわてずに、Z'sSTAFFの「ペン」メニューの中のマスク反転アイコンを2回使ってください。それで正常になるようです。

〈EFFECT〉

サブメニューウィンドウが開いて、その中に4つアイテムが並んでいます。それらのうちひとつを選択してください。するとウィンドウが消えます。エフェクトをかけたい領域をマウスで指定してください。エフェクトをかけたい矩形領域の左上と右下にマウスカーソルを持っていって　左クリックしてください。その中に選択したエフェクトをかけます。指定を間違えたときは右クリックでやり直せます。メニューに戻るときは、1点も指定しない状態で右クリックしてください（要するに、どんな場合でも2回右を押せば帰ってきます）。

白黒処理（MONOTONE）と微分処理（DIFFERENTIAL）については説明の必要はないでしょう。一度試せば理解できます。問題は残りの2つ、ランダムフラクタル処理（RANDOM FRACTAL）とフレア処理（FLARE）です。この2つには、パラメータがあるのです。

〈RANDOM FRACTAL〉

囲んだ領域を自然なランダムさで（なんしゃそりゃ）崩します。ランダムフラクタル処理は　グラデーションのような　滑らかに変化する画像に対して威力を発揮します。縦にグラデーションをかけた部分を囲んでみてください（アルゴリズムの性質上横方向に変化するグラデーションを処理しても意味がありません）。雲のような画像が現れるはずです。

さて、メニューウィンドウのRANDOM FRACTALと書いてある右に、数字の入った枠があります。これがパラメータです。この値は、フラクタルで画像の崩れる度合いを示したものです。ここでマウスを左または右クリックすれば　パラメータの値が増減できます（左で＋、右で－）。値の取りうる範囲は0から9のあいだです。値が大きいほど崩れ方も派手になります。

〈FLARE〉

光源にしたい部分をマスクしておき（これは前述のマスキングペイントでもいいしZ'sSTAFFのマスキングでもいいのです）その周りを囲みます。すると　マスク領域の周辺に　その光源の色が淡く滲み出てくるはずです。そこでマスクを解除すると（これにはZ'sSTAFFのマスク解除命令を使ってください）光源がぼうっと光っているように見えることでしょう。

フレアにもランダムフラクタルと同様に、ウィンドウ内にパラメータの入った枠が2つあります。フレアの場合パラメータは2つです。左のパラメータは明るさ　右のパラメータは広がりぐあいを表しています。

フレアは　However、描かれるまで多少時間が掛かると思います。理想をいうなら　光を滲ませるアルゴリズムを理解したうえで使うのがいいのです（気分としてもっとも近いのは「ぼかし」処理です）。こつとしては　内側のマスクより大きめに範囲を指定することがまず挙げられるでしょう。でないと　光の広がりが途中で切れます。これはみっともないものです。また、マスク部分の形や大きさによっても、フレアの様子はかなり変わってきます。うまく使うと面白い効果が出せますので、しばらく使ってみてぜひとも体験してください。

＊

ものによっては　非常に実行が遅い機能もあります。そんな機能はたいてい　右ボタンを押しているうちに実行を中断して戻ってくるようにしてあります。

例によって　ウィンドウ左上のクローズアイコンでメインメニューに戻ることができます。

第4章　おわりに

今回は難しい話は抜きにしました（おいおい本当かよ）。無事に使えるようになりましたか？

メモリが1Mバイトの方は残念してください。ただ　Z's EXはZ'sSTAFFから起動しなくてもエフェクタとして使えるプログラムですから　うまいプログラムを作って名前をSTAFF68K.Xにしてしまえば、1Mバイトでも動くようになるかもしれません。気が向いたらフォローします。

しつこいようですが　Z'sSTAFFのウィンドウをすべて閉じてからZ's EXを起動してください。そこそこ使えるツールだと自負しております。ぜひとも活用してください。

あと、操作性や機能で気にいらないところ　ほかにこんな機能もほしいといった要望などございましたら、お知らせいただけると幸いです。

最後に、このプログラムを実現する可能性を開いてくださった、電脳倶楽部とPIC FILERの山田前示さんにお礼を申し上げたいと思います。

TOOLS

グラフ作成ツールMichael

Ueno Kazuhiko
上野 和彦

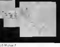
ディスクでしか掲載できないちょっと大きな複雑作業　グラフ作成ツールMichaelです。「マイケル」と呼んでください。基本的なグラフ作成機能に加え　グラフィックツールの要素を盛り込んでいます。なお、マルチウィンドウですがSX WINDOW上のソフトではありません。ご注意ください。

いちいち紙にグラフを描いてデータの傾向を見るのは面倒だ　という先輩用のあなたにピッタリのソフトを作ってみました。片対数/両対数用紙もサポートした簡易グラフィックエディタつきのグラフ作成ツールMichaelです。マルチウィンドウの採用で画面上でグラフを紙のように扱えます。グラフの印字品質もまあまあで　市販のグラフ作成ツールのものと比べても見劣りしません。3次元グラフとか棒グラフといった凝ったグラフは表示できませんが　私たちがふつうにグラフ用紙にX軸Y軸をとって描くようなグラフについては自信を持っておすすめできます。

ただし、プログラムの性格上　プリンタを持っていないとほとんどありがたみはありません。CZ系の24ピン（または48ピン）プリンタをお持ちの方　実験のデータ整理などにぜひお役立てください。

Michaelとは

このツールは「人間がグラフを描く動作をパソコンでシミュレートする」ということをイメージして作りました。私たちが日常グラフを描くときの動作を思い起こしてみましょう。まず　座標軸をとり、対応する点を打ってその上から定規で直線を引いたり　自由定規で曲線を引き　テンプレートで文字を入れていきます。このツールはこれとまったく同じ手順でグラフを描いていくものです



これがMichael

X　Yの座標を入力すると点を打ってくれて　そのあとグラフィックエディタで好きなところに線や文字が入れられます。このため、説明力豊かなグラフが作成できます。どんな強力なツールを作っても結局最後は人間の手で文字や図を入れたくなるんですよね。

以下にMichaelの特徴をまとめてみましょう。

1）　マルチウィンドウの採用で同時に最大4つまでのグラフが扱える
2）　グラフィックエディタがついているので　簡単に文字や直線　曲線が入れられる
3）　最小2乗直線の計算ができる
4）　キーが豊富である
5）　印字の際は画面イメージの1.5倍のフレームを取るので印字品質がよい
6）　各グラフには説明がつけられるため、あとで見てなんのグラフかすぐにわかる
7）　グラフの表示範囲を自動的に決定する（もちろん　あとで修正できる）

使用方法

Michaelを使用するには少なくともプリンタドライバと数値演算ドライバが組み込まれた状態で、Michaelのあるディレクトリに移動したうえで、

A＞MICHAEL WORK¥

のように起動してください。

付録ディスクの場合　ディスク2のMichael.BATのアイコンをダブルクリックすれば起動できます（ただし印刷などは不可能）。

ウィンドウ部の基本操作

● ウィンドウの移動

ウィンドウの上側についているタイトルバーを1回クリックして　移動したい位置でもう一度クリックします。

ビジュアルシェルやSX WINDOWのようなドラッグ操作ではありませんので注意してください。このツールではマウスをドラッグして操作するということはありません。あと、クリック操作は軽快に行ってください。これさえ覚えておけば　あとはだいたい勘でわかるでしょう。

● ウィンドウ内のスクロール

グラフ用紙を大きくしたときスクロールバーが現れます。非常に細いので注意して見てください。そこをクリックするとウィンドウの中身をスクロールさせることができます。

その他の操作については図を参照してください。

コマンド解説

起動時に表示されるメニューウィンドウに沿って各コマンドを解説していきます。コマンドを実行するときはメニューウィンドウの各ボタンをクリックしてください。

1）　データの入力

グラフにプロットする点の座標を入力するコマンドです。メニューウィンドウの <データの入力> というボタンをクリックするとテーブルウィンドウが開きます。1から4までのうちどれかを選択すると対応する表ウィンドウが開きます

図2を見てください　たとえば　（1 0，2，0）（1 1　2，1）（1 2，2 2）（1 3　2 3）の4点をグラフにプロットしたい場合、まずX座標だけを横1行に入力して、次にY座標のみを別の行に入力します。

リターンキーを押すとメニューに戻るので注意してください。場所の移動はカーソルキー（コントロールキーならさらによい）を使用してください。キー操作は、

CTRL＋E　上
CTRL＋D　右
CTRL＋X　下
CTRL＋S　左
CTRL＋H　バックスペース
CTRL＋M　リターン
CTRL＋Y　1行カット

CTRL+L　1行ペースト
CLR　初期状態に戻す
となっています。

X座標をどの行に書いたかをMichaelに通知しなければなりません。X座標を書いた行をクリックするとキーウィンドウが開くので、ここからXのマークを選びます。同様にプロットするデータの記号を選択してください。

2) グラフ用紙

表示するレンジ、グラフの大きさ、目盛りの種類などを決めます。

データ入力後、メインメニューから<グラフ用紙>を選択し、適当に設定するとグラフが表示されます（グラフ用紙設定から抜けるにはメインメニューまたはデータ入力ウィンドウをクリックしてください）。データの入力が不完全な場合はグラフは表示されません。注意してください。

■目盛り

対数設定ができます。Xを普通にYを対数にすると（逆でもいいんですけど）片対数グラフになります。目盛りと書いてある位置のボタンを押すと表示が交互に切り替わります。目盛りを変えると表示レンジも変えなくてはいけないのでレンジも自動設定します。また、座標軸に0または負の数があると対数に設定できません。

■最大値、最小値

グラフの目盛りの最大値と最小値です。最初は自動設定された値になっています。対応するボタンをクリックすると入力ウィンドウが開くのでそこで入力してください。キーボードがローマ字や全角のモードになっていると設定できません。

■数字間隔

数字を表示する間隔です。対数目盛りのときには10と1しか設定できません。1にするとすべての目盛りに数字を打ちます。10にすると10目盛りおきに数字を打ちます。

■目盛り間隔

目盛りを刻む間隔です。対数目盛りのときには10と1しか設定できません。

■説明

X軸、Y軸の説明です。対応するボタンをクリックすると入力ウィンドウが開きます。日本語FEPを組み込んでいるときは日本語入力もできます。

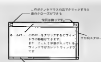
■幅

グラフの表示ドット数です。この数字を変えると印刷したときのグラフの大きさも変わります。

3) キーテーブル

キーテーブルの表示位置を決めます。ウィンドウ外に設定すると表示しません。

図1



ネームバーの左右にあるマークは、Michaelでは飾りにしかついてません。表を閉じると、それに対応するグラフも同時に閉じるようになっています。

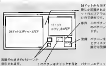
スクロールバーは、グラフウィンドウが大きくなったときに現れます。小さいので見落とさないようにしてください。ウィンドウの大きさは、今回はMichaelに固定させて設定にしました。

図2



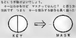
		1	2	3	4
X	x_data	1 0	1 1	1 2	1 3
○	y_data	2 0	2 1	2 2	2 3
●	y1_data	3 0	3 1	3 2	3 3

図3



キーの作画もマスクの作画も操作は同じです。24ドットのエディットエリアでパターンを描いて、16ドットに変換して、16ドットエディットエリアで変換後のパターンを修正するという手順がよいでしょう。

キーはわかるけど、マスクってなんだ？　と思う方もいるでしょう。マスクは、キーの型紙です。つまり、キーを表示する部分を黒く塗ったものです。



4) ロード

表をロードします。ファイルウィンドウにファイル名が表示されるので目的のものをクリックしてください。

5) セーブ

表をセーブします。複数の表を開いているときには、どの表をセーブするかを選んでください。

6) 削除

ファイルを削除します。ファイルウィンドウで削除するファイルを選んでください。

7) キーの作成

グラフに点をプロットするときのキーを作成します。

8) マスクの作成

グラフに点を表示するときのマスクを作成します。

9) 終了

Michaelを終了します。

10) TOOL

グラフをエディットするときのモードを決めます。

■エディットモード

直線、曲線、文字から選んでください。

■線種類

エディットモードで直線、曲線を選んだとき描画に使用する線の種類を決めます。線の太さが2つと実線、点線、1点鎖線、2点鎖線の4つの線種、合計8パターンが使用できます。が、画面上では線の太さは区別がつきません。線の種類で区別することをおすすめします。

■文字種類

エディットモードで文字を選んだとき描画に使用する文字の大きさを決めます。大文字と小文字のどちらかを選んでください。

●グリッド

カーソルの動きを8ドットおきにします。一定間隔で文字を入力するときや平行線を引くとき便利です。

11) EDIT

グラフをエディットします。表が複数開いているときはどの表のグラフをエディットするか聞いてきます。また、グラフウィンドウをカレントにした場合もこのモードとなります。

12) DEL

エディットで入力した図、または15)、16)で入力した式などを消します。消したい線や文字をクリックしてください。このコマンドから抜けるときはグラフウィンドウの外をクリックしてください。

13) カット

エディットで入力した図、または15)、16)で入力した式などをすべて消します。

14) ペースト

13)でカットしたデータを復帰します。

15) 最小2乗

表に登録されているすべての点に対して最小2乗法によって回帰直線を求めます。Y=aXは化学系の検量線の作成などに便利だと思います（必ず原点を通らなければならないときはY=aX+bが使えないんですよね）。

Y=bEXP(aX)は片対数グラフの直線です。この型はいろんな式に出てくるのでそれらの定数を求めるのに便利です（グラフも表示されているので、はずれたデータを除いて最小2乗を取ることができます）。

また、このときの線種はTOOLで指定したものとなります。

16) 式の入力

限られたかたちしかありませんが式を入力してグラフにできます。

17) 属性

最小2乗で計算した定数などを表示させるときに使います。グラフの特定はDELとほぼ同じです。

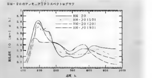
18) 印刷

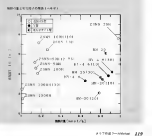
グラフをプリンタに出力します。用紙はB5を想定しています。プリンタのA4の位置に設定してください。

*　　*　　*

それでは皆さん、ぜひこのツールを活用してください。

図4





SX-WINDOW開発セット&アクセサリプログラム

解説 泉 大介
資料提供／シャープ株式会社

SX-WINDOWの内部に興味がある人、プログラムを組みたいと思っている人には、なくてはならない技術資料、ツール類、ライブラリを一挙公開です。誌面には絶対載せられない膨大な量ですが、ぜひとも活用してください。いくつかのサンプルプログラムも楽しめます。

ディスク3のアクセサリ

ディスク3のSX_samplというディレクトリにはSX-WINDOW上で動くちょっとしたアクセサリプログラムが入っています。このうち、SXLIFEは中森氏が本誌の記事用に作成したもので、そのほかはシャープから提供されたサンプルプログラムです（SX-WINDOWのチェックと遊びを兼ねて作成したものだとか）。お手元のウィンドウで遊んでみてください。

●SXLIFE

SX-WINDOW用のライフゲームです。それぞれの青い球の周りに何個の青い球が存在するかで、その青い球が次の世代に生き残れるか、死んでしまうか、それとも新しい青い球が誕生するかどうかが決まります。青い球が織り成すパターンの面白さを味わってみてください。なお、マウスの右ボタンをクリックすると、マウスカーソルの位置に青い球が出現します。また、青い球の上でクリックすればその球を消すことができますので、自分でパターンを組んでその変化を楽しむこともできます。

●SXeyes

目玉がマウスカーソルを追いかけるというサンプルです。ウィンドウシステムには定番のアイテムですね。メモリに余裕のある方は、目玉を10個も表示すれば一種異様な雰囲気に浸ることができます。ぜひお試しを。

●15パズル

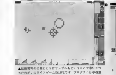
4×4のマスに入っている15個のタイルに番号が振ってあります。それを1から順に並べ替えてください。16個のマスに15個のタイルですから1マスの空きができます。空きマスと同じ行、あるいは同じ列のタイルの上で左ボタンをクリックすると、クリックされたタイルが空き升の方向に押しやられるようにして移動します。こうしてタイルを移動しながら数字を揃えていくわけです。

●パス名

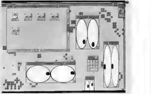
ファイルのフルパス名を表示するウィンドウを開くサンプルです。フルパス名を表示したいファイルのアイコンを、ドラッグしてこのウィンドウに放り込んでください。「クリップボードへ全体を転送する」と書かれたボタンをクリックすると、フルパス名がクリップボードに転送されます。転送されたパス名は、ノートなどのプルダウンメニューで取り出すことができます。このプログラムはもともと、ノートでCONFIG.SYSを書き換えるために作られたものでした。

＊

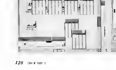
本体メモリが2Mバイト以上の方は、なにも問題はありません。SX-WINDOWを起動したあとディスク3、ディスク4を挿入し、サンプルプログラムをダブルクリックしてください。本体メモリが1Mバイトしかない方は、メインメモリ1Mバイトで起動するSX-WINDOWのシステムディスクの作り方を別項で紹介していますので、1Mバイト用のSX-WINDOWシステムディスクを作成してサンプルを実行してください。



▲技術資料の公開とともにサンプルをということで急いで作ったのが、このライフゲームSXLIFEです。プログラムは中森章さんにお願いしました



▲ライフゲームをクリ動き、15パズル、時しるの間、目玉もよ目しているぞ。



▼15パズルは、何回いくらでも解けるから、たくさん画面に表示しておいて片っ端から解くというのも暴力的でいい



◀パス名は、アイコンを放り込むとパス名を表示し、クリップボードに取り込める。長いフルパス指定をするのに便利。

ディスク4の内容

ディスク4はSX-WINDOW用アプリケーションの開発ディスクです。これらはシャープから提供されたものを編集部で構成したものです。ある程度の技術的知識を持った人を対象とした開発者向けの資料ですから、十分な理解は困難かもしれません。本誌では、今回の特集をはじめ、今後ともSX-WINDOWの解説を行っていく予定です。また、くれぐれもメーカーに直接問い合わせることはご遠慮ください。

SXディレクトリの中には、開発ディスクのファイル内容を説明するREADME.DOCと、6つのサブディレクトリが収められています。README.DOCは都合上ディスク3に収めたアクセサリプログラムの説明も兼ねており、README.DOCでQuickStartディレクトリ内のファイルとして説明されているのがディスク3のSX-Sampleディレクトリに収められているプログラムです。なお「とけいX」は入っていません。ご了承ください。

このディスク4に入っているファイルは大きく3つに分けられます。ひとつは開発のための資料、もうひとつは開発のためのツール類、そしてSX-WINDOW用のサンプルプログラムです。このディスクはSX-WINDOW用のアプリケーション開発の補助ディスクであり、これ単体でアプリケーションが作れるというものではありません。XCあるいはアセンブラ、リンカが別途必要となります。すでに発売されているこれらの開発言語を、SX-WINDOW用のアプリケーションが開発できるように拡張するものと思っていただければいいでしょう。

サンプルプログラムについて

初めに、ディスク4に入っているサンプルプログラムの簡単な使い方を紹介しておきます。

●GRSAMP

グラフマンの機能を使って、画面に図形や文字を表示する小さなサンプルです。

■TSSAMP

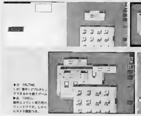
ウィンドウを開いて文字を表示するという、最も基本的なプログラムです。SX-WINDOWプログラミングの基礎を学ぶにはちょうどいいサンプルだと思います。

■TMSAMP

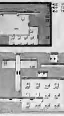
コマンド実行用のウィンドウです。COMMAND.Xと同じようにコマンド名をキーボードから入力し、リターンキーを押してください。このウィンドウではCOMMAND.Xのコマンドを入力しても実行することはできません。

■DBLTIME

ダブルクリック間隔の短さを競うゲームです。最も短い時間がハイスコアとして表示されます。ちなみに編集部での最短タイムは0.03秒です。逆に、ダブルクリックの間隔の長さを競うゲームとしても遊べます。



▶左　DBLTIME
いかに素早くダブルクリックできるかを競うゲーム
▶右　TMSHELL
簡単なコマンド実行用のウィンドウです。しかもヒストリ機能つき。



◀左　GRSAMP
◀中　TSSAMP
◀右　TMSAMP

こちらはハイスコアは残りません。

このプログラムはウィンドウ間の通信サンプルとして用意されたものです。複数のDBLTIMEを起動すると、スコアはそれぞれのウィンドウで独立ですが、ハイスコアは通信によって他のDBLTIMEのウィンドウにも伝えられ、常に最速のタイムが表示されます。

■DBLTIME2

DBLTIMEをリエントラント（再入可能）に書き直したものです。

■TSHELL

TMSAMPの機能を拡張したものです。こちらはHuman68kのコマンドも実行することができ、さらにヒストリ機能を備えています。コマンド入力の履歴はカーソル上下キーでたどることができます。また、カーソルキー、BS、DELキーなどを使って、以前入力したコマンドを再編集して実行することも可能です。

SX-WINDOWに関する資料

開発資料の中でまず目を通したいのは、DOCディレクトリに収められたファイルです。ビジュアルシェル上（VS3.X）でもダブルクリックで読むことができますが、積極的に活用するにはED.Xで読み込んだり、プリンタに打ち出すとよいでしょう（約100ページになりますが）。また、WP.Xのファイル入力でも読めますが、GRAPH.DOCなどはファイルサイズが大きすぎて一度には読み込めません。

ここに収められたファイルはSX-WINDOW上のアプリケーションの動きを制御する各種マネージャを理解するための資料です。低レベル（ハードウェアに近いレベル）のマネージャから順に並んでいます。高レベルのマネージャのほうがアプリケーションと直接関わり合うのですが、SX-WINDOWで使われる用語の理解のためにも最初から読み進んだほうがいいでしょう。

SX-WINDOWの各マネージャとのやりとりは、SXコール（Human68kのDOSコールのようなもの）を利用して行います。このSXコールのリファレンスマニュアルが収められているのがREFディレクトリです。SXコールはここにマネージャごとにまとめられています。ファイルの拡張子がREFとなっていますが、DOCファイルと同様に内容は普通のドキュメントファイルですので、ED.XやWP.Xで読むことができます。

ここには、SXコールの一覧を収めた拡張子がLSTのファイルも入っています。DOC、REFファイルはすべてプリントアウトすると400ページにも及ぶ大作ですが、LSTファイルはコンパクトですのでプリントアウトして綴じておくと便利です。

開発を支えるファイル群

SX-WINDOW用のプログラムはアセンブリ言語、またはC言語で作成します。アセンブリ言語を使用する場合は、INCLUDEディレクトリのSXCALL.EQU、SXCALL.MACを利用します。これらのファイルにはSXコールのエントリとマクロが定義されています。

C言語を利用する場合は、INCLUDEディレクトリに入っているヘッダファイルと、LIBディレクトリに入っているライブラリを使用します。SXコールと該当するCの関数はREFファイルで対にして解説されています。

TOOLディレクトリに入っているSXKERNELは、SX-WINDOW本体であるSXWIN.Xからファイル操作やディスク操作といったデスクトップ機能を省いたものです。これはSAMPLEディレクトリに入っているTSSAMPなどをコマンドラインから実行すると起動されます。プログラム開発途中での「実行→チェック」用に用意されたものだといえるでしょう。

TSSAMP.Xを実行するにはFSX.Xがメモリに常駐している必要があります。手持ちのHuman68kのCOMMAND.Xから実行したい場合は、SX-WINDOWのSYSディレクトリに入っているFSX.Xをまず実行します。これでFSX.Xがメモリに常駐して、サンプルプログラムを実行可能となります。常駐を解除するには、もう一度FSX.Xを実行してください。サンプルプログラムにはSXKERNEL.XではなくSXWIN.Xが必要なものもあります。

同じくTOOLディレクトリに入っているDB.Xは、SXコールに対応したデバッガです。SXコールに対応するAラインファンクション（一種のおまじないのようなもの）をファンクション名でデバッグできるので非常に便利です。前述のSXKERNEL、SXWDBは、このデバッガに対応している点がSXWIN.Xと異なっています。

SX-WINDOWの資料とツールが配布され、やっとSX-WINDOW用のアプリケーションを作成する下準備ができました。SX-WINDOW用プログラムの作り方が特集でも触れられていますので、ディスクの資料を参照しながら読んでみてください。

1MマシンでSX-WINDOWを動かす

1MバイトのX68000でSX-WINDOWを起動する方法を以下に示します。

●手順1　SX-WINDOWのコピー

ビジュアルシェルやコマンドモードなどでSX-WINDOWのコピーディスクを作ってください。以下の作業はコピーされたディスクを使って行いますので、オリジナルディスクは再び保管しておいてください。

●手順2　ファイルのリネーム

まず（コピーされた）SX-WINDOWのディスク内のファイルのうち、大きなメモリを必要とするファイルの名前を付け替えてしまいます。SYSディレクトリに入っているASK68K.SYSとSHELLディレクトリに入っているHEMWIN.Xの2つのファイルを、適当な名前に変えてしまってください。これらのファイルが占有しない分だけフリーエリアが増えますから、これでSX-WINDOWが起動できるようになります。ただし、漢字は使えません。

■手順3　さらにフリーエリアを増やす

変更したSX-WINDOWのディスクをドライブ0に差し、リセットボタンを押してください。SX-WINDOWが起動します。カレンダーなどが表示されていることと思いますので、右上のXアイコンのところで左ボタンを押し「全クローズ」を選択して一旦すべてのウィンドウを閉じます。

Aドライブのアイコンをダブルクリックすると表示されるウィンドウの中からCONFIG.SYSを探してください。見つかったらこれをクリックして反転表示にし、次に「ノート」を開きます。ノートにCONFIG.SYSの内容が表示されていますね。ノートの中央あたりにある

```
DEVICE=¥SYS¥PRNDRV.SYS
DEVICE=¥SYS¥RSDRV.SYS
DEVICE=¥SYS¥OPMDRV.SYS
```

の3つの行の先頭に、＊を付け、saveボタンを押してCONFIG.SYSをセーブしてください。

これはSX-WINDOWから「プリントアウト」「RS-232C」による通信」「FM音源の演奏機能」をカットする指示です。

ノートのsaveボタンを押してCONFIG.SYSをセーブしたら、再びリセットしてSX-WINDOWを起動してください。FM音源が鳴らない、漢字が使えないなど機能面に制限ではありますが、これで1Mバイトのメモリでも支障なくサンプルプログラムを実行できるSX-WINDOWシステムディスクができあがりました。ただし、相変わらずメモリは厳しいので、SXsymを2つ起動するようなことはできません。個々にお試しください。

今回のサンプルにはありませんが、FM音源が必要な場合は再びノートでCONFIG.SYSを読み込み、OPMDRVの行の先頭に付けた＊を除ってリセットすればOKです。ただし、その分フリーエリアは減りますので、もしかすると目的のプログラムが実行できない可能性があります。

ウイルス検出プログラム DOCTOR2.X

編集部　プログラム協力：中森　章

DOCTOR2.XはX68000上のウイルスを治療または予防するワクチンプログラムです。使用上の注意事項をよく読んでお使いください。また、旧バージョンのDOCTOR.Rをご使用の方は今回のDOCTOR2.Xに差し換えてお使いください。

DOCTOR2.Xは、X68000のSRAMに潜伏するウイルスに対応するワクチンプログラムです。本誌で報告されているX68000上のウイルスはいずれもSRAMに入り込むことで感染するというもので、フロッピーディスクに記録されるIPL部分を書き換えることで、次々と別の本体に感染していく性質を持っていました。そこで、このDOCTOR2.Xは主に次のような機能を持たせています。

1) フロッピーディスクのIPL領域を検査、および治療する。
   DOCTOR2
2) 本体のSRAMを初期化する。
   DOCTOR2 -P
3) SRAMにDOCTORを常駐させる。
   DOCTOR2 -T

●なお、DOCTOR2 -? などでオプションスイッチのヘルプが出ます。

それでは、基本的な使用法について具体的に解説しておきましょう。

フロッピーディスクの検査と治療

ドライブ0に挿入されたフロッピーディスクのIPLを検査します。手持ちのシステムディスクを起動し、調べたいフロッピーディスクをドライブ0に挿入してください。

DOCTOR2.Xの入ったディスクをドライブ1に入れ、

A>B:DOCTOR2

と実行してください。異常がなければ

IPLをチェックしています。セルフチェックOK
これはHuman純正のIPLです。
ドライブ0のHuman は正常です。

と表示されるはずです。

ただし、ゲームソフトなどの一部でオリジナルのIPLを使用しているものもあります。その場合には

これはHuman純正のIPLではないが大丈夫でしょう。
シフトキーを押すと続行します。

となります。

もしも、ウイルスに感染するなどの原因で正常なIPLが書き換えられている場合には画面が一瞬赤くフラッシュし、

これはウイルスに冒されています!!!

治療しますので、ドライブ1に健康なHuman Ver 2.0(最新版)を入れて
リターンキーを押してください。

といったメッセージを表示します。治療というのは冒されたIPLの部分に正常なIPLをコピーするということです。ただし、問題となったいわゆるFORCEウイルスの場合にはIPLのバージョンが書かれた部分も書き換えられるため上記のメッセージが"Human Ver Pro"などとなってしまいます。純正のIPLはHuman68kのバージョンによって、1.0と2.0の2種類がありますが、いずれのバージョンでも特に支障なく治療できるはずです。メッセージにしたがって安全なシステムディスク（マスターディスクなど）をドライブ1に入れてリターンキーを押してください。

ドライブ0の Human は健康になりました。
よかったね。

となればOKです。ここで、治療しようとしたディスクにライトプロテクトシールが貼られていたりすると、

ドライブ0の書き込みに失敗しました。(書き込み禁止?)
プロテクトシールをはがしてドライブ0にディスクを入れ直し
リターンキーを押してください。

と表示されます。再度確認してください。

さて、注意してほしいのは、IPLが純正ではなくオリジナルのIPLの場合です。もともとプロテクトシールが貼ってあるゲームソフトの場合ならほぼ安全ですが、カーオリジナルIPLを使用したディスクがウイルスに感染した場合、DOCTOR2.Xでも治療はできません。ウイルスは殺すことができますが起動できなくなってしまいます。

SRAMを初期化する

2番目の機能は、SRAMの初期化です。ウイルスに感染したフロッピーディスクでシステムを起動すると、ウイルスはSRAMに入り込み、X68000が感染した状態になります。そこでSRAMのプログラムエリアを初期化するには、コマンドモードから、

DOCTOR2 -P

としてください。

これで、SRAMのプログラムエリア($ED0100～$ED3FFF)が初期化されます。ただし、SRAMを使用中の場合、内容が消えてしまうのでご注意ください。

また、ウイルスの感染によってシステムがうまく起動しなくなったときは、OPT.1キーを押しながらシステムディスクを起動させSRAMの初期化を行ってください。

DOCTORをSRAMに常駐させる

上記の2つの機能で、ウイルスに感染したディスクやマシンを治療することができます。しかし、できることならウイルスへの感染は未然に防ぎたいものです。

そこで、DOCTORにはあらかじめSRAMに常駐させて起動時のチェックを行う機能があります。コマンドモードから、

DOCTOR2 -T

としてください。これでDOCTORがSRAMに転送されます。

以後、フロッピーディスクで起動するたびに自動的にディスクのIPLを調べ、上記のDOCTOR2を実行した場合と同様のメッセージを出すようになります。

ドライブ0にディスクが入ってないと、フロッピーの挿入を待つメッセージを出します。

ドライブ0に Human のディスクを入れてください。 5

ハードディスクがつながっている場合にはカウントが0になるまで待ってハードディスクからの起動に切り替わります。が、最初からハードディスクで立ち上げたい場合にはシフトキーでカウントダウンを中止してハードディスクから起動します。

なお、DOCTORを外したい場合には、

DOCTOR2 -P

で、SRAMを初期化するだけでOKです。

ヘルプメッセージ

```
B>doctor2 -?
X68k IPL Checker V2.01 Copyright 1990 (Oh!X 中森章)

使用法 doctor [オプション]
オプション /t   SRAMへの転送を行う
         /h   ブレークキー[illegible]
         /     [illegible]を初期化する
         /     プログラムエ[illegible]を初期化する
         /     SRAM [illegible]を初期化する
         /     リザーブエリア[illegible]を初期化する
         /     DOCTORを実行する
なお、/t 以外は[illegible]ブレークキー[illegible]チェックを行います
```

補足解説

今回のDOCTOR2.Xは1990年6月号で発表したワクチンDOCTOR.Rを新たに作り直したもので、機能の拡張、エラーチェック強化、SCSIへの対応、新たなウイルスへの考慮などさまざまな点で改良されています。

前回のDOCTOR.Rでは、SRAMをクリアするzap.xと、DOCTOR.RをSRAMに転送するtfr2.xが別のプログラムになっていましたが、今回はすべての機能をDOCTOR2.X 1本にまとめました。また、7月号でお知らせしたように従来のDOCTOR.RはSCSIを採用したX68000 SUPER-HDでは使用できません。もちろん、今回のDOCTOR2.XはSCSIへも対応しています。

さて、DOCTORはチェック用として内部に持つIPLのセルフチェックを行いますが、これがNGとなる場合があります。これはSRAMの一部がビット化けしているためで、機種によってはハードウェア的にSRAMの状態が不安定なケースがあるようです。そこで、今回のDOCTOR2.Xではその場合も想定して、NGとなった際の修復機能も付加しています。ただし、あまり頻繁にNGとなるようでしたら、編集部宛にご連絡ください。

なお、SRAMはいくつかのエリアに分かれており、オプションスイッチによって初期化するエリアを指定できます。必要に応じてご使用ください。

X68000用
めぞん一刻より暁に鐘は鳴る他2曲

Saitou Akiyoshi
斎藤 彰良

X1/turbo用
涙で綴るパパへの手紙

Sasaki Kouji
佐々木 孝司

あけましておめでとうございます。寒い日が続いていますが、皆さんかぜなどひいてませんか？ さて、新年初のLIVE in '91はあの「メゾン一刻」のTVサイズミュージックと、バービーボーイズです。きっとアニメファンにもロックファンにも満足していただけると思います。

響子さぁ～ん，好きしぁ～～～！

新年早々とんでもない見出しを付けてしまいました。通の人でなくてもわかるかもしれませんね。Live in '91を飾る栄光の1曲目は、めぞん一刻から「暁に鐘は鳴る・93話TVサイズ」です。ちょっと短めの曲ですので、打ち込むのも楽なのではないでしょうか。物足りないと思う人もいるかもしれませんね。そんな人のためにオマケでもう2曲付けてしまいましょう。曲は「響子の悲しみ」と「夜の雨」です。OPMAやOPMDでバリバリのサンプリング音を駆使した作品が多いなか、異色ともいえる落ち着いた雰囲気をかもし出しています。元気いっぱいの作品と比べるとちょっと物足りない気もしますが、たまには静かな音楽にひたるのもよいのではないでしょうか。

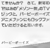
この作品たちはプリセット音を使っているので、ほかの音楽プログラムなどを演奏させたあとで聴くと音色が変わっているかもしれません。注意してください。

作者の斎藤君にいわせると、これらの作品のなかでは1曲目の「暁に鐘は鳴る」が本命なのだそうです。チャンネル数の不足のために、ノーブでコードになるべきところがコードになっていなかったり、いくつかの点で完璧とはいえないそうですが、納得できるデキだとか。「響子の悲しみ」はちょっと悲しい感じのする曲で、第27話の「消えた惣一郎?」の夜中にかかった曲なんだそうです。「夜の雨」はピアノ曲です。聴いてみると、確かにタイトルの感じがよくわかる気がします。私の勝手な解釈を書いておきましょう。夜の公園の噴水の側にたたずむ2人。水面を静かに叩く雨。黙りこんで気まずくなった2人をときにやさしく包みこんでくれる――。う～む、考えただけでもロマンチックですねぇ。今月号の発売日はクリスマスのちょうど1週間前ですから、この記事を読んでからイブを迎える人もいることでしょう。"がんばってくださいね（声 島本須美）"

めぞん一刻



バービーボーイズ



こちらもキョウコさん・バービーです

さて、今月のX1はMusic BASIC用の「涙で綴るパパへの手紙」バービーボーイズです。このバンドは男女のツインヴォーカルで構成されています。女の人の名前がキョウコさんなのはただの偶然です。これで男の人の名前がコーエンさん（この時期禁句）だったらね――（本当はコンタさんといいます）。

作品はアルバム「Listen」からの選曲です。比較的に古いアルバムで、バービーがバリバリに流行っていたころのものです。私も「Listen」はLP版を持っていたので、楽しみに聴いてみたのですが、どうも聴き覚えがないのです。でも確かに作者の佐々木君のコメントにはそう書いてある。う～む、と考えてみると、CDのほうにはオマケに1曲ついていたのではないだろうかという疑問にぶちあたります。調べてみると、確かにありました。CDバージョンの7曲目ですね。うむ、まいった。

さて、佐々木君は同時に4曲を送ってきてくれました。知名度としては断然上のものである「ごめんなさい」や「泣いたままでlisten to me」などもあったのですが、トータル的にはこの掲載作品がいちばんよくまとまっていたと思います。ただし、全体的に原曲と比べるとテンポが遅く、モタつき感がありますので、テンポを上げてみてはいかがでしょう。具体的には50行の"T145"を"T153"に直してください。これで原曲とほぼ同じになります。聴き比べるとわかると思いますが、モタつき感は減っています。

それと、2小節ごとに遅れる感じがありますので、対策を練りましょう。たとえば、ほかのコマンドにくらべて音色を設定する作業はMMLにとってかなり大変なものになっています。そこで、@コマンドを同時にいくつもやらないで、前の小節の最後の音を若干短めにして、そこに@コマンドを入れてやると、MMLがオーバーワークになる可能性が減ります。MMLは割り込みを禁止して作業をしますので、オーバーワークになるとテンポずれの直接の原因になってしまいます。楽譜どおりに打ち込むよりも、原曲どおりに音が鳴るように心掛けましょう。

(S.K.)

リスト1 暁に鐘は鳴る

[illegible]

リスト2 囃子の悲しみ

```
[illegible]
```

リスト3

日本音楽著作権協会（出）許諾第9072083-001号

```
[illegible]
```



```
[illegible]
```

リスト4 雨で書るパパへの手紙


```
[illegible]
```

```
[illegible]
```

電話番号が変わります

Ogikubo Kei 荻窪 圭

平成3年1月1日 あなたの電話番号が変わります。東京都以外の人にはあまり関係ないけど，当事者たちにとっては大問題。だって，電話帳やらなんやらを変更しないといけませんからね。というわけで 簡単ながらも便利なプログラムの登場です。

今回はいきなり番外編である。まあ こんなこともあろう。

さて，NTTという企業がある。ダイヤルQ²で運げているNTTだ。Q²ってのはたいていエッチなことに使われている。NTTさんは企業とかがビジネスで使う用途を想定したそうだが 馬鹿である。伝言ダイヤルを見ていれば Q²がどう使われるかなんて明らかではないか。これはもう確信犯である。そんでもって，伝言ダイヤルに飽きたらなくなった人はダイヤルQ²を始め 電話代が30万円とか100万円とかになったりするのだ。若者の心の隙間から生き血をすするNTTと呼んであげよう。さすが 民間企業になって利潤を追求するようになるとやることが違う。JRも見習ってスケベ列車でも走らせる（もちろん表向きはビジネス利用ということにするのだ）といいかもしれない。

さて そのNTTさんのいうことには 平成3年 つまり1991年という21世紀の10年前という記念すべき年に入ったとたん 東京03局の市内局番を4桁にしようというのだ。元旦の午前2時にである（どうして2時かは知らないけれど）。これまでも 3桁では足りないってことで新しい加入者をどんどん5で始まる4桁の市内局番にしていたのだが（Oh!X編集部も4桁） それでもまだ足りないらしく，古い人も一斉に4桁にしてしまう計画らしい。どうして古い人まで4桁にする必要があるかというと，177-2533ってかけてもちゃんと天気予報にかかるようなもの ってとこから類推してください。細かくは綿愛。

東京の人口はそんなに増えなくても 電話の加入者は増えるのだ。会社の数は増えるし 会社は回線を増やすし，FAX専用回線なんて引いたりするからだ。そういうわけで 番外編は時事ネタの局番変更。『大人のためのX68000』には格好のネタではないか。

そこで考えた。局番変更に向けて，どうするのが一番楽にすむだろうか。Kamekaseのマクロでできるか。CARD PRO 68Kではどうか。予算表はいやだし。うーん。

結論。あまりに安易なところへ落ち着いてしまってもうしわけないが X BASICを使うことにした。X BASICなら

1) 誰もが持っている
2) テキストファイルならどんな形式でもOK

というメリットがあるからである。

プログラミングへの道

出来上がったのは簡単なファイル入出力のプログラムである。入力ファイルひとつに出力ファイルひとつという一番単純なパターンだ。入力ファイルから1レコード読んでは加工して1レコード書き出す ってのをファイルがなくなるまで繰り返すだけだ。1レコードってのはまあ 1行のこと。Human68kでいう1行ってのは（MS-DOSでも同じだが），CR+LFを区切りとした1文字列だ。

ちなみに，CRってのはキャリッジリターンのことで 日本語でいうと"復帰"である。どこに"復帰"するかというと，行の先頭だ。LFってのはラインフィードのことで，日本語では"行送り"または"改行"である。だからCR+LFを復帰改行という。2つあわさって初めて一般でいう"改行"が成り立つのだな。タイプライター文化の名残といっていいだろう。

では，もうちょっと細かいアルゴリズムを考える。

まず最初に，NTTのパンフレットを見る。"現在3ケタの市内局番の前に『3』をつけ，4ケタの市内局番になります"

これだけ。簡単そう。

頭の中で整理する。"『03』があったら市内局番を取り出して，3桁だったら頭に『3』を付加する"だけでいい。

これをプログラムできるところまで分解する必要がある。

1) 入力ファイルの形式を決める。

これはテキストファイルならなんでもOKとしよう。

2) 出力ファイルの形式を決める。

これは入力ファイルに準ずる。

問題は "3" を付加したために その行が1桁増えてしまうという現実をどうするかってことだ。私はご存じのとおり 怠慢である。そこで『気にしない』ことにした。CSV形式なら 項目長を無視できるので気にしなくていい。区切りなしASCIIだと困るが 区切りをタブコートにしているなら きっと 問題ない。区切りありASCII形式なら項目長を無視できるので 問題ない。ということで 困題はないのだ。

3) アルゴリズムを決める。

まず『03-』を探す。これだと住所の項目や郵便番号の項目に『103 55』なんてのがあったり 電話番号が『052 403 9988』なんてのがあったりしてもマッチングしてしまうが，これも気にしない。次のチェックがあるからだ。

『03』の次の4桁を取り出す。この4桁目が『-』かどうかで市内局番が3桁か4桁か判断できるから。郵便番号や市内局番に『03』が含まれていても その4桁後に『-』がくることは（おそらく）ない。住所に含まれているときも 『X03-XXX-XX』などというケースはきっと稀だ。

ここで この連載の精神が明らかになる。つまり『たった1回しか使わないプログラムだから 簡潔であるべし。余計な気遣い

は無用)。

今年から来年にかけてしか使わないものに、手間暇かけるのは無駄というものだ。予定外のマッチングなんて滅多にないのだから。そういうケースはあとから手作業で直したほうがずっと早い。少なくとも、そういった稀なケースのための処理をプログラムに追加するよりは、だ。

第1回「大人のためのX-BASIC入門」(?)

そんなこんなでリストのプログラムが出来上がった。縁起ものなので説明はする。

変更するデータファイルであるが、ファイル名を入力させるのも間抜けなので、プログラム中に書いてしまった。変数宣言部のFN (File Nameの略ね)に書いてある。ここを自分に都合のいいものに変更する。

続いて、NFNには拡張子をOLDにしたファイル名が入る。

ちょっとややこしいが、"TEL.DAT"てなファイルをこのプログラムにかけると、"TEL.DAT"を"TEL.OLD"にリネームし、"TEL.OLD"を入力ファイルに、"TEL.DAT"を出力ファイルにする。こうすると、見かけ上、TEL.DATが書き換えられて、バックアップファイルとしてTEL.OLDが作られたように見えるのだ。

入力されたファイルは1行ずつ"R-BUF"(リードバッファの略ね)という変数に読み込まれる。R-BUFはstr型で宣言しているので、MAXでも255文字である。これ以上長い行には対応していない。これはひとえに私の怠慢であるが、そのあたりを柔軟にすると、プログラムが長くなるのでやめた。だから、データファイルにはCSV形式など1行がコンパクトにおさまる形式を勧める。

読み込まれたらまず"03-"があるかどうかチェックし、なければその行を書き出す。write時、行の末尾にCR+LFを付加すること。

とりあえず、"03-"を見つけたら次の4桁をチェック。必要なら変更して、そこまでを書き込み用のバッファである"W_BUF"(ライトバッファの略ね)へ追加する。そして、残りの部分からまた"03-"の検索を開始する。

元のファイルから1行ずつR_BUFへ読み込んで、必要なら加工しながらW-BUFへ代入し、1行分終わったらファイルへ書き出すという作業をレコードがなくなるまで続けるわけだ。

ただこれだけだが、repeat—untilでなく、while~endwhileを使いたいという理由により、レコードのreadが2カ所にある。whileの判定にEOFかどうか(End Of File、つまり、ファイルが終わりかどうかをチェックすること。freads関数のリターンコードでわかる)を使っているため、whileの前に一度readしておく必要があるのだ。

まあ、こういう書き方もあるのだと思っていただければいい。私は諸般の事情により、こういう考えに馴れてしまったのだ。

全レコードの処理を終えたら、クローズして終了する。

不親切な本プログラムの使い方

なにはさておき、更新したいデータファイルを用意する。Kamikazeならカルクシートから標準テキスト形式で保存すれば得られる。CARD PRO-68KやCYBER NOTE PRO-68KならコンバートCSV形式にコンバートする。電話番号は"XXX-XXX-XXXX"のようにハイフン"-"で局番を区切ってあること。"XXX(XXX)XXXX"とか"(XXX)XXX-XXXX"ってな書き方には対応していない。どうしても括弧を使いたい、って人は、プログラムを書き換えてもらいたい。簡単だから。もちろん、「03」を省略してある悲惨なものにも対応していない。

用意できたら、エディタ(X-BASICで打ち込んでも構いませんが)でこのリストを打ち込む。ファイル名はちゃんと自分の使うデータファイル名に直すこと。

そんでもって、セーブし、BASICからloadコマンドでロードする。

ここで実行するわけだが、注意すべきは"このプログラムは必要最小限の処理しかしない"ということである。ファイル名を間違えたとか、1行あたり300バイトもあったなんてケースには対応していない。たとえば、リネームする関数はリネームする予定の名前と同じ名前のファイルがあるとそこで止まってしまうが、だからといってこのプログラムは何もしないのだ。そういったエラーが出て止まったら、自分で対処してほしい。

冷たいが、エラー処理のためにプログラムが長くなるのととちらがいいか、っていわれると、私は「たった1度や2度しか使わないプログラムで100行程度なら短い方がいい」と答えるのだ。長くなるとそれだけ打ち間違いも増えるわけだし。

そういうわけで、最初は4、5行の短いテスト用データファイルを用意して実験することを勧める。

うまくいかなかったときでも、拡張子がOLDのファイルにちゃんと元のデータが残っているので、こいつを元のファイル名に戻してやり直せば大丈夫だ。

そういったわけで、CIコンパートしてコンパイルするのは勧めない。止めたほうがいい。動く保証もしない。

なお、こういういい加減なプログラムでファイル処理を行うのは本来危険なので、データファイルのバックアップは取っておこう。

テスト用のファイルでうまくいったら、平成3年の元旦の午前2時を待つ。そして、午前2時になったら「えいやっ」と実行する。うまくいけば新しいアドレス帳で新年を迎えることができるわけだ。

完了したら、また元の形式(KamikazeなりCARD PRO-68Kなり)に戻して、何事もなかったかのようにその住所録を使えばいい。それだけである。

これにて「大人のためのX68000」番外編はおしまい。次から「大人のためのX-BASIC入門」が始まる、てなことは(きっと)ないからご安心を。

BASIC的トラブルシューティング

打ち間違いやらなにやらで、うまくいかないことは多い。そういうときはBASICのメリットをフルに生かしてごまかそう。

1) 実行途中で止めてしまったり、エラーで止まったりした場合

ファイルがopenしたままになっていると思われるので、

FIN()

をまず実行しよう。FIN()ってのはファイルをクローズする部分(終了処理)の関数名だ。BASICはこうして特定の関数だけを実行できるので、便利である。

そして、デバッグする。

2) 同じファイル名があるのでリネームできない、っていわれた場合

よくあるケース。こいつも、

kill NFN

でいい。これは元のファイル名にデータがあって、拡張子OLDのものがじゃまな場合だ。エラーで止まったときは変数の内容は保持されているので、NFNには拡張子がOLDのファイル名が入っている。

拡張子.OLDのものを元の名前に戻したいときは、

kill FN

で元のファイル名を消して

frename(NFN,FN)

を実行すればいい。BASICの利点だ。

動作確認は私が使っている電話帳ファイルによって行っているが、本文で書いたようにうまくいかないケースは多々ありうる。そういうときはごめんなさい。

リスト

```
10 /**************************************************************/
20 /*        1991年からの03局番変更対応プログラム                  */
30 /*                                   90/11/08  K Ogikubo      */
40 /**************************************************************/
50 /**************************************************************/
60 /*            変数 定数定義                                   */
70 /**************************************************************/
80 str FN[64]="B:TEL.DAT"                  /* DB file name       */
90 str NFN[64]                             /* 拡張子がoldのもの  */
100 str R_BUF[255]                         /* Read Buffer        */
110 str W_BUF[255]                         /* Write Buffer       */
120 str TOKIO[4]                           /* 市内局番           */
130 str FIRST[255]                         /* ポインターより前   */
140 str LAST[255]                          /* ポインターより後ろ */
150 str CRLF[2]                            /* CR+LF              */
160 int RC=0                               /* 汎数リターンコード */
170 int FRC                                /* Fileリターンコード */
180 int R_PTR                              /* Record Pointer     */
190 int RB_LEN                             /* Read Buffer Length */
200 int COUNTER=0
210 CRLF=chr$(13)+chr$(10)                 /* CR+LF              */
220 /**************************************************************/
230 /*          メインプログラム                                  */
240 /**************************************************************/
250    INIT()                              /* 初期処理           */
260    while not(FRC)
270       R_PTR=instr(1,R_BUF,"03-")       /* 03-を探せ          */
280       if R_PTR=0 then {
290          W_BUF=R_BUF                   /* 03-はなかった処理  */
300       } else {
310          while R_PTR
320             TOKIO=mid$(R_BUF,R_PTR+3,4)
330                            /* 市内局番の取り出し             */
340             if right$(TOKIO,1)="-" then {
350                FIRST=left$(R_BUF,R_PTR+2)
360                LAST=right$(R_BUF,RB_LEN-(R_PTR+2))
370                W_BUF=W_BUF+FIRST+"3"+TOKIO
380                            /* 市内局番に3を追加              */
390                print TOKIO," を 3"+TOKIO," に変更"
400                COUNTER=COUNTER+1
410             } else {
420                FIRST=left$(R_BUF,R_PTR+6)
430                LAST=right$(R_BUF,RB_LEN-(R_PTR+6))
440                W_BUF=W_BUF+FIRST
450                            /* そのまま新しいバッファへ       */
460             }
470             R_BUF=LAST                  /* 旧バッファ更新    */
480             RB_LEN=strlen(R_BUF)
490             R_PTR=instr(1,R_BUF,"03-")
500          endwhile
510          W_BUF=W_BUF+R_BUF
520       }
530       RC=fwrites(W_BUF+CRLF,FNO2)
540       FRC=freads(R_BUF,FNO1)
550       W_BUF=""
560       RB_LEN=FRC
570    endwhile
580    FIN()                                /* 終了処理          */
590 end
600 /**************************************************************/
610 /*      初期処理(ファイル名変更  ファイルオープンなど)        */
620 /**************************************************************/
630 func INIT()
640    int SRC
650    FNCHG()                              /* ファイル名 oldの作成 */
660    FNO1=fopen(NFN,"r")
670    FNO2=fopen(FN,"c")
680    FRC=freads(R_BUF,FNO1)               /* 第1 read          */
690    W_BUF=""
700    RB_LEN=FRC
710 endfunc
720 /**************************************************************/
730 /*    ファイル名変更処理                                      */
740 /**************************************************************/
750 func FNCHG()
760    int RC
770    str PERIOD="."
780    RC=instr(1,FN,PERIOD)
790    if RC=0 then {
800       RC=strlen(FN)+1
810       FN=FN+PERIOD
820    }
830    NFN=left$(FN,RC)+"OLD"
840    print FN+" を "+NFN+" にリネームします。"
850    frename(FN,NFN)
860 endfunc
870 /**************************************************************/
880 /*          終了処理                                          */
890 /**************************************************************/
900 func FIN()
910    print "修正した電話番号は ",COUNTER," 件です。"
920    fclose(FNO1)
930    fclose(FNO2)
940 endfunc
```

宇宙要塞CADの逆襲 その2

プロジェクトチームDōGA　MAX田口

前回に引き続き、紙飛行機2号の作成です。分量が多かったので2回に分かれてしまいましたが、今回でいよいよ完成です。前回のおさらいはちゃんとできていますか？ 覚えていない方は、1990年11月号を手元に置いて読んでください。

はじめに

皆さん、宇宙要塞CADは我々（私？）の想像以上に強固だった。残念ながら、11月号だけでは、とても陥落させることができなかったのだ。どうしても分量が多くなってしまって、2回に分けるしかなかったわけだ。おかげで、今月も原稿を書くはめになった。これは、Oh!X編集部の陰謀か、はたまた独断か、まだのさしがねか、それとも私が悪いのか？

しかし、こうなったらじたばたせずにさっさとCADでもなんでも攻略して、ナイアスでもクリアしようっと。

前回のおさらい

それじゃあ、さっそくCADを立ち上げて、前回作った「紙飛行機2号」をロードしよう。前回は、ページの都合上セーブの仕方について書けなかったけど、ちゃんとセーブしておいてくれたかな。もしセーブしないでそのまま終了した人がいたら、その人は、残念だけどもう一度最初から作り直すしかない（まあ、そんな人はいないと思うけど……）。ロードの仕方は、FileModeのloadのところをクリックして、ファイル名を入力するだけだ。ファイルのロードが終わると画面に、何か表示されるけど、今回は何もせずに、「固定終了」を選択する。これだけで、ファイルのロードは終わりだ。

さて、ロードが終わったからといって、いきなり「Edit Mode」に入った人は、いないかな。CADを立ち上げたら、真っ先にするのは、面を作ることじゃなくて、そう、Cパネルの固定変更だ。今回は2度目だから変更する点だけを書いておくから、変更の仕方を忘れた人は、11月号を見るように。PANEL1の変更点は、表示スケールを「8」にすること、gridを「200」にすること、アトリビュートの指定を「body」にすること、の3つだ。そして、次にPANEL4の3Dカーソルの移動幅を「100」にする。これが済んだら、再びCパネルの表示をPANEL1にして、設定は終わりだ。

さて、アトリビュートの指定をするところで気がついた人もいるかもしれないけど、前回たくさん指定したアトリビュートが「body」だけになっている。なぜかというと、前回では、アトリビュート名が「body」の面しか作らなかったので、前回登録したはずのアトリビュート名が消えてしまった（というかセーブされなかった）のだ。そこで、残りのアトリビュート名を再び入力しないといけない。

アトリビュート名の登録は、Attribute ModeのAttr.登録で行うんだけど、覚えていたかな。Attr.登録をクリックしたら、今回の「紙飛行機2号」で使う「body」以外の4つのアトリビュート名を順番に入れていこう。「cockpit」、「engine」、「wing」、「missile」と、4つのアトリビュート名をキーボードから順に入力し終わったら、ESCキーを押してアトリビュート登録を終了しよう。

これで、前回セーブしたときと同じ状態になったので、安心して制作の続きができるぞ。



図1
簡易度別の面の例

胴体部分の作成の続き

<その4>簡易度3の面の作り方

胴体部分は、11月号でほとんど完成している。残っているのは、機首と胴体を結ぶくびれた部分だけだ。これは簡易度3の面を使って作る。簡易度3の面といっても作り方はほとんど簡易度2の面と変わらない。

さっそく、Edit Modeに入って、アトリビュート名が「body」になっているのを確認したら、面を作り始めよう。さっきもいったとおり、いまから作るのは簡易度3の面といっても簡単なほうなので、全然難しくない。だから、前回の復習だと思って作ってもらえばいい。

今度作る面は、側面図を主に使って作る。まず3Dカーソルを、（−400, 200, 100）のところに持っていく。この点が第1頂点になるので、スペースキーを押して、点を確定する。第1頂点を決定したら、次に3Dカーソルを下に1と2分の1マス分（−300）動かす（座標は（−400, 200, −200）になる）。ここが第2頂点になる。今度は、3Dカーソルを、右に1マス分（+200）、上に2分の1マス分（+100）動かす。そして、今度はマウスカーソルを正面図のところに持っていって、3Dカーソルを左に2分の1マス分（−100）動かす（ちゃんと座標が（−200, 100, −100）になったかな）。ここが第3頂点になる。あとは、マウスカーソルを側面図に戻して、3Dカーソルを上に1マス分（+200）だけ動かして（−200, 100, 100）にしたあと、第4頂点を確定して、面を確定すればいい。これで、胴体のくびれた部分の右側の面は完成だ。ほら、簡単だったろ？ これで、面の作り方を思い出したかな？

あとは同じようにして、残り2つの面を作ればいいわけだ。

＊

・四角形　第1頂点（−200, −100, −100）
　　　　　第2頂点（−200, −100, 100）
　　　　　第3頂点（−400, −200, 100）
　　　　　第4頂点（−400, −200, −200）

簡易度3　胴体のくびれた部分の左側の面、側面図を中心にして作る

・四角形　第1頂点（−400, 200, −200）
　　　　　第2頂点（−400, −200, −200）
　　　　　第3頂点（−200, −100, −100）
　　　　　第4頂点（−200, 100, −100）

簡易度3　胴体のくびれた部分の底の面、上面図を中心にして作る

＊

さて、最後の面の辺はすべて、いままでに作った辺と重なったはずだ。こうなると、本当にここの面を作っていないのか、それとも実際にはここに面があるのかわからなくなる（図3参照）。箱を作って、ふたの面を作るときには、必ずこういう状態になるので、比較的よくお目にかかる状態だと思う。今回は、面の数が少ないので、ここはまだ作ってない面だとか、ここは作ったとかわかると思うけど、面数が多くなるとすぐにわからなくなる。こういうときは、次のようにしたらいい。

まず、ふたになるはずの面の頂点の位置に3Dカーソルを移動させる（もちろんその頂点は、前に作った面の頂点と同じになるはずだ）。今回だったら（−400, 200, −200）ぐらいでいいだろう。

そして、Eパネルの上から4行目の「点−>面」というところをクリックする。この「点−>面」には、現在の3Dカーソルの位置を頂点にしている面を順番に呼び出す、という機能があるので、ここを何回かクリックして、いまから作ろうとしている面があるかどうか確認すればいい（図3参照）。

<その5>コックピットの作成

さて、胴体の作成が終わったら、次はコックピットを作る。わずか5面しかないし、どれも簡易度1～2の面ばかりなので、すぐにできるはずだ。ただ、いままで作ってきた面のアトリビュート名は「body」だったけど、これから作る面のアトリビュート名は「cockpit」なので、アトリビュートの指定の変更を忘れないように。



図2　正面図

＊

アトリビュート名「cockpit」

・長方形　第1頂点（100, −100, 100）
　　　　　第2頂点（100, 100, 100）
　　　　　第3頂点（−300, 100, 200）
　　　　　第4頂点（−300, −100, 200）

簡易度2，コックピットの前の面，正面図を中心にして作るとラク

・長方形　第1頂点（−300, −100, 200）
　　　　　第2頂点（−300, 100, 200）
　　　　　第3頂点（−400, 100, 200）
　　　　　第4頂点（−400, −100, 200）

簡易度1　コックピットの天井の面，上面図を使って作る

・長方形　第1頂点（−400, −100, 200）
　　　　　第2頂点（−400, 100, 200）
　　　　　第3頂点（−500, 100, 100）
　　　　　第4頂点（−500, −100, 100）

簡易度2　コックピットの後ろの面，正面図を中心にして作るとラク

・四角形　第1頂点（−500, −100, 100）
　　　　　第2頂点（100, −100, 100）

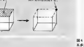


図3　面の確認方法

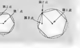


図4
正多角形の作り方

第3頂点(−300 −100 200)
第4頂点(−400 −100 200)

簡易版1、コックピットの右側の面、側面図を使って作る

・四角形　第1頂点(−500 100 100)
第2頂点(100 100 100)
第3頂点(−300、100 200)
第4頂点(−400 100、200)

簡易版1、コックピットの左側の面、側面図を使って作る

＊

さて、これで胴体部分の作成が終わったことになる。簡易版1、2の面が作れるようになったら、これだけでかなりいろんな物体が作れるはずだ。次からはいよいよコマンドを使った面の作り方について説明する。ここまでちゃんとできた人は、一度セーブしてから次に進むこと。

エンジン部分を作る

<その1>正多角形の作り方

いままでは、ほとんど前回の復習みたいなものだった。でも、これからは違うぞ。これから作る面は、いよいよAパネルのコマンドを使って面を作るのだ。コマンドを使うので、すべての頂点を指定する必要がないぶん操作はラクだ。でも、慣れないうちは、自分の考えていたのと違う形になったりすることもある。それに、コマンドを使って面を作ると、格子点上に頂点がこないので、あとでその点に3Dカーソルを移動するとき面倒臭いので多用は禁物だ。

最初は、正多角形の作り方だ。これは「紙飛行機2号」のエンジンの部分になる。だから、アトリビュートもちゃんと「engine」に変更してから、次を読むように。

まず、正多角形を作るには、正多角形の作成モードにしないといけない。Aパネルの下から4行目の「正多角形作成」のところをクリックすればAパネルが正多角形の作成モードになる。今回作るのは、正六角形なので、まずAパネルのいちばん上の行の「角数 12」と表示してあるところで、マウスの左ボタンを6回クリックして角数を「6」にする。次に3Dカーソルを(−500 300 100)の点に移動させる。ここで、スペースキーを1回押す。もちろんスペースキーの代わりに、Bパネルの「点指定」をクリックしてもいい。この点が正多角形の中心、正確にいうと正多角形に外接する円の中心になる。そして、3Dカーソルを正面図で左に2分の1マス分(−100)だけ移動させて、もう一度スペースキーを押す。この点は、正多角形の大きさ、これまた正確にいうと、正多角形に外接する円の半径になる。また、正多角形の頂点がこの点にくるように作られる(図4参照)。そしてさらに、3Dカーソルを正面図で上に1マス分(+200)だけ移動させて、スペースキーを押す。この点は、実をいうと(今回は)正面図上の点だったらどこでもいいのだ。この点の役目はただひとつ。中学校のときに平面を決定するには最低3点が必要だと習ったと思う。つまり、最後の1点はできる正多角形がどの面の上にあるかを決定するためのものだったのだ。この点を適当に変えると、傾いた正多角形も簡単にできるようになっている。3つの点が確定したら、リターンキーを押す。すると、図4のような正六角形が表示されるはずだ。そして、正多角形ができたら、ESCキーを押して、正多角形作成モードを終了する。

<その2>角柱の作り方

次は、さっき作った正六角形を底面とする角柱を作ることにしよう。まず、Aパネルの上から5行目の「角柱作成」のところをクリックして、角柱作成モードに入る。そして、3Dカーソルを(−500,200 100)に移動させて、スペースキーを押す。そして、次に3Dカーソルを側面図で左に3と2分の1マス分(−700)移動させて、もう一度スペースキーを押す。このとき、座標は(−1200,200 100)になっているはずだけど、もしなっていなかったら、指定を取り消してから正しい点で点指定すること。そして、ちゃんと2つの点を指定したら、リターンキーを押す。すると、さっき作った正六角形を底面として、高さがいま指定した2点の距離となるような角柱ができるはずだ。失敗した人は、いちいち面を全部消すか、セーブしたところからやり直すしかない。

角柱作成で指定した2点は、角柱の高さと傾きだということはすぐにわかったと思う。もちろんこれを使うと、

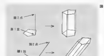


図5　角柱の作り方

ビサの斜塔なんかも簡単に作ることができる。指定する直線をちょっと傾かせればいいだけだ(図5参照)。

ただ、この角柱作成で注意する点は、底面の指定だ。角柱作成モードでは、底面を作るとき、底面を指定するということができない。そこで、角柱作成では、底面は現在の面ポインタがある面になるのだ。つまり角柱を作るには、あらかじめ底面を作っておかないといけないということになる。そうするとあとは、高さのベクトルを指定するだけで、側面と上面を作ってくれるのだ。

さて、さっきの解説で聞き慣れない言葉が出てきたと思う。面ポインタなんか「そういやマニュアルに載ってたな」ぐらいしか覚えてない人がほとんどだろう。ポイント面と、各コマンドの関係は、CADを使ううえでとっても重要なことだけど、今回は完璧に理解なんかしなくていい。詳しくはマニュアルを読めばいいとして、とりあえずは次のように覚えておけばいいのだ。「1面を作るとその位置に面ポインタが移動するので」普通、面ポインタは最後に作った面にある。だからさっきは、正六角形を底面とした角柱ができたわけだ(最後に作った面がエンジン部分の正六角形だったからね)。そこで、角柱を作るときは、底面を作って、すぐに角柱を作るようにして、底面の作成と角柱の作成を、ワンセットでやるようにする。そうすれば、面ポインタなんかまったく気にしなくていいぞ。

あとは、同じ角柱を正面図で下に1と2分の1マス分(−300)だけ移動したところに作ればエンジンの部分は出来上がりだ。まず、中心が(−500 200 −200)で、前に2分の1マス分(100)動かした点を頂点に持つ、正面図に平行な正六角形を作る(長い文章だなぁ)。そのあと、すぐに高さが3と2分の1マス分(700)の角柱を作るのだ。これが終わったら、もう一度セーブしてから次に進もう。

柚姫の明るい悩み相談室(番外編)

D-CGA転覆す!?
バージョンアップサービスにトラブル発生!

1990年9月に行われたCGAシステムのバージョンアップサービスにおいて、重大なトラブルが発生したとの情報を得た柚は、さっそくDoGAプロジェクトルームに潜入。これより柚の恐怖の追及が始まるのであった。

＊

ピ、ピー、静かなマンションにブザーの音が鳴り響く。プロジェクトルームのドアが静かに開いた。

柚「いったいどのようなトラブルだったのですか?」

MAX田口(以下M)「知らん、知らん」

柚「大きなバグでも発見されたのですか?」

M「知らんゆうてるやろ!」

柚「まさかウイルスでは?」

M「あほなこというな。確かにバグの問題もあるけど、あれは発送上のトラブルや。それ以上は何もいえん」

MAX田口氏はそういって中に入っていった。柚はマイクを片手に、プロジェクトルームの奥深くへとさらに進んでゆく(BGM、インディ・ジョーンズのテーマ)。

柚は事件当日の発送担当者がFFE三島氏であることをつきとめ、インタビューに成功した。

FFE三島「俺の責任じゃないよー。コピーの担当者が悪いんだ。コピーの担当は島さんだよー」

どうやら、発送のトラブルとはコピーミスらしい。次に柚はroot島氏をつかまえた。

柚「コピーの担当者が悪いんだ、という声が聞かれているのですが」

root島(以下r)「つまり、今回のバージョンアップサービスで2枚のディスクが送られたんやけど、その2枚がまったく同じやったという苦情がたくさん来たんや」

柚「なぜそんなことが起きたんでしょうか」

r「さあ、なんでかさっぱりわからんなー。まったく謎の仕組としか……」

(むこうでかまた氏とお話しき人のくしゃみが聞こえる)

柚「どの程度の苦情が届いたのですか?」

r「1〜2割だったから、重大な話やな。でも、コピーミスの苦情だけやないで。なんでもMAILに、このバージョンはディスク3枚組だ、とか"IOCS.Xの新しいバージョンが入っている"とか、ウソばっかり書いてあって、だいぶ混乱を招いたんや」

どうやら、トラブルの根は深いようだ。さらに、マスターディスク作成担当レイバー小林氏にインタビューをした。

柚「マスターディスクに問題があったそうですが……」

レイバー小林(以下レ)「エ〜、そーなんかなぁ」

柚「はぁ」

レ「システムが2枚とサンプルが1枚で、バージョンアップしたのはシステムなんやから間は空ってるやん。IOCSははじめ入れるつもりやったんやけど、シャープにいわんと、勝手にでけへんからなぁ」

(聞自すればいいことなのにと思う柚)

柚「今後はIOCSを入れていくんですか」

レ「入れるつもりやけど、シャープにまだ電話かけてないから、なんともいわれへんなぁ」

(やっぱりさっさと電話すればいいのにと思う柚)

レ「どっちにしてもあのバグから比べたら、たいした問題とちゃうやん」

どうやら、大きなバグもあったようだ。バグ出し担当のGAVAN重田氏を追求してみよう。

柚「バージョンアップ版に大きなバグがあるそうですね」

GAVAN重田(以下G)「FFE(モーションデザインツール)がぜんぜん動かないみたいやね。HANIM(ハイスピードアニメーション)は動くけど、そのために必要なCRB(色数削減ツール)がバグってて、画面の下のほうに黒い帯が発生するそうだ」

柚「それって、ほとんどバージョンダウンっていいません? 何かコメントお願いします」

G「バグ出しはやったんだけど、バグ出しするあとからあとからバージョンアップしたので、追いつかなくて……(悲痛な叫び)。だいたい土壇場になってコンパイラ変えたりするか?」

どうも今回の事件は責任がはっきりしない。責任者であるかまた氏に話を聞いてみることにした。

柚「いったいなんでそんなことになったんですか」

かまた(以下か)「わたしゃ、ずっと旅行しとったから、今回はノータッチなんやけどなぁ。いや〜、スイスの山はやっぱりきれいやな。朝焼けのマッターホルンは感動ものやで。夜明け前から山登って……」

柚「そんな話は聞いていません。それで、今後の対策というのは?」

か「やっぱり、バージョンアップサービスのバージョンアップサービスを近日中に行わないかんやろなぁ。ディスク1枚を無料で送るつもりやわ」

柚「わかりました。責任者として、今回のトラブルについてどう思われますか?」

か「このような遺憾な事態となったのは、まったくイカンなぁ」

柚「……ありがとうございました」

●

このたびはたいへん混乱を引き起こし、申し訳ありませんでした。上記の内容は冗談(妙にリアリティのある冗談?)として、関係者一同深く、深〜く反省しております。バージョンアップサービスのバージョンアップサービスがお手元に届くまでいましばらくお待ちください。また、9月から1月にかけて、今回のトラブル以外にも、試験や学祭があったため、トラブルに関する問い合わせや要望にぜんぜん対応できなかったことをお詫びします。

それにしても、当プロジェクトルームには毎日多数の問い合わせが来ます。「CGAシステムはまだ手に入りますか?」などというのはかわいいもので、なかには、とっくに締め切られている前回のCGAコンテストのビデオのお金を勝手に振り込んでおいて、「まだ来ない!」と苦情をいうような困った方もいます。基本的に、当方に問い合わせて返事がなかったらDoGA内でトラブルが発生して、対応できない状態に陥っているのだと好意的に受けとめてください。できるだけのことはやっているのですが、まぁ今年、留年した5人のスタッフに免じて、お許しを……。

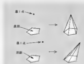
各読者通達事項

●CGAコンテスト事務局より

12月15日の新聞自伝でも御しらわ〔りきかったから誰も気づかなかっただろうなぁ〕ますます盛り上がるCGAコンテストの審査員が新たに2名加わりました。映像作家の金井俊雄氏とアニメーション作家の古川タク氏です。これで審査員は豪勢11名という豪華さです。

さて、締め切り(12月31日)まで残すところあと2週間足らずとなったわけですが、皆さんラストスパート頑張ってますか? "完成しそうにないから、また来年のコンテストを目指そう"などと思ったら大きな間違いですよ。そんなことを考えていると、作品はいつまでたっても完成しません。来年になれば、コンテストのレベルも上がるし、出しネタ〈作品〉に1年以上つき合っていると、嫌になって、完成せずに次の作品を手がけてしまうものです。与えられた期限を守るというのは、作品制作の重要なポイントです。必要ならば、ストーリーを短くしたり、もっと簡単にできる表現に切り替えるなどの対応も必要となるでしょう。あとはもう根力と根性で、画面の頭がなるまで、毎日の精進、頑張ってください。

図6　角錐の作り方



翼の部分を作る

<その1>翼を作る

翼の部分は　難易度2の面を2つ作るだけなのですぐにできると思う。ただ、何度もいうようだけどアトリビュートの変更だけはお忘れなく。

＊

アトリビュート名「wing」

・四角形	第1頂点(−1000	1000	400)
	第2頂点(−700、	1000	400)
	第3頂点(−500	400	100)
	第4頂点(−1000	400、	100)

難易度2　左上の翼。側面図を中心にして作るとラク。また　Y座標の移動は、この面は上面図に入りきらないので正面図を使ってするといい。

・四角形	第1頂点(−1000	1000、−500)
	第2頂点(−700	1000、−200)
	第3頂点(−500、	400、−200)
	第4頂点(−1000	400　−500)

難易度2　左下の翼。側面図を中心にして作るとラク。また　Y座標の移動は　この面は上面図に入りきらないので正面図を使ってするといい。

<その2>角錐の作り方

最後にミサイルの部分を作る。まずアトリビュートを「missile」に変更したら　正面図に平行な面を作る。

＊

・正方形	第1頂点(−1100	1000	400)
	第2頂点(−1100	1100	500)
	第3頂点(−1100	1100	500)
	第4頂点(−1000	1000、	400)

難易度1　超簡単な面。これがこれから作る角錐の底面になる。

＊

角錐の作成では　角柱のときと同じで面ポインタのある面（つまり直後に作った面）が底面になる。ただ　角柱のときと違うのは　指定する点が1点　つまり角錐のいちばん上の点（なんていうのか忘れた）を指定するだけで　角錐ができるということだ（図6参照）。この場合　点を確定するつもりで　うっかりスペースキーを押してしまうと　有無をいわさずそこを頂点とする角錐ができてしまう。間違えた点でスペースキーを押してしまったら、いちいち面を消さないといけない。だから　角錐の作成は十分注意して行ったほうがいい。

今回は　先ほど作った面を底面とした角錐を作る。まずは、Aパネルの「角柱作成」の下にある「角錐作成」のところでマウスをクリックして　角錐作成モードに入る。そして、3Dカーソルを（100 1000 400）のところに移動させて　リターンキーでも　スペースキーでもどちらでも好きなキーをどちらか1回だけ押す。これだけだ。これだけで、あっという間に角錐が出来上がるのだ。もっとも、これはちゃんとポインタ面がさっき作った面にあったらの話で　もし変なことをして　ポインタ面が移動していたら、予想もつかない面（というかポインタ面がある面）を底面とした角錐ができる。だから、角錐を作るときも　角柱を作るときと同様に　底面の作成と角錐の作成はワンセットでやったほうがいい。

あとはもうひとつ作るだけだ。まずは　底面を作る。

＊

・正方形	第1頂点(−1100	1000、−500)
	第2頂点(−1100	1100　−500)
	第3頂点(−1100	1100、−600)
	第4頂点(−1000	1000、−600)

難易度1　超簡単な面。これが左下の角錐の底面になる。

＊

そして　面を作ったら、すぐに角錐作成モードに入って頂点を（100 1000 −500）とする角錐を作る。これが終わったらセーブして　そして　CADを終了する。

ついに完成だ

いままでに作ったのは　「紙飛行機2号」の右半分だけだった。でも　この「紙飛行機2号」は　左右対称なので、あとはMIRRを使って左右反転コピーすれば右半分の部分が自動的に作られて、「紙飛行機2号」が完成する。MIRRでは　オプションをつけずに反転コピーすればいい。ただ、万が一間違えたときのために　出力ファイル名は　元のものと違う名前にしておいたほうが無難だ。MIRRを使ったら　もう一度CADを立ち上げて　見てみよう。ちゃんとできていたら成功だ。今回はCADの特集なので　これで終わりだ。できたら、これを使って　かっこいいカットを作ってほしい。もし、かっこいいカットができなくてもモデル（紙飛行機2号）のせいにはしないように。



図7　完成時の形状ファイル

おわりに

CGAシステムが使えないという人の約8割は　CADでつまずいているといわれている（当社推定）。CADに慣れるには　1時間でも多くCADを使うことだ。今回作った「紙飛行機2号」なんか、慣れると5分ほどでできてしまう。実際私はこれを5分ほどで作ったのだから間違いない。でも、その5分でできる物体を作るのを説明するのに　いったいどれだけかかったか──。

ここ数カ月　隔月連載といったわりには毎月連載をしているような気がするが、次回は　かまたさんの"実用的な構造体の使い方「闘えロボット君2」"をお送りできるはず、だろう。

モデラー高津のLOGIN

「好きやねんCG大阪会議」のレポートの中にも少し話が出ていましたが　DoGAでは現在、オリジナルのウィンドウシステム「Ko-Window」を開発しています。なぜ　CGAの団体であるはずのDoGAがウィンドウシステムを開発しているかというと　単に趣味の世界です。しかし、来年完成予定のDoGA CGAシステムVer 3は、このKo-Window上で動作させるかもしれません。J&PのBBSでは　常にKo-windowに関する最新の情報をアップしていますので　ぜひご覧ください。

Ko-Windowの利点と欠点は以下のとおりです。

利点

一仕様が公開されている。

一プログラムをCで記述できるので　開発が容易である

欠点

一動作が不安定。

一ころころとバージョンが上がっている。

一イベント駆動型（注1）なので　プログラムが組みにくい

現在、Ko-Windowでは以下のようなアプリケーションが動いています。

HELLO.WIN	ご存じハローワールド
DCLOCK.WIN	デジタル時計
PED.WIN	パターンエディタ
FINDER.WIN	ファイルの選択および移動を行うプログラム
ACLOCK.WIN	アナログ時計
CALC.WIN	電卓
X.WIN	Xのロゴを表示する（注2）
EYES.WIN	マウスカーソルを見つめる目玉（注3）
BACK.WIN	背景を変える
COMMAND.WIN	ウィンドウ版Command.X
POPUP.WIN	ポップアップメニューを実現する
WED.WIN	テキストエディタ
FREE.WIN	メモリの空き容量の表示するプログラム
SLIDE.WIN	CGAシステムのPICファイルを表示する
GRAPH.WIN	グラフィック表示のデモ
REND.WIN	CGAシステムのレンダラー
MAP.WIN	メモリの使用状況を表示するプログラム

なお　このKo-WindowはJ&P以外のネット（注4）でも少しずつ　公開していきます。

注1

普通のCのプログラムでは、main関数から順番に必要な関数を呼び出していきます。Ko-Window用のプログラムでは　関数の呼び出しをウィンドウシステムに任せてしまい　用事ができたら関数を呼んでねっ　といってmain関数を終了します。すると、ウィンドウを描き直すとか、キーボードから入力があったとか、なにか関数が必要になったらその関数を呼んでくれるわけです。普通のプログラムはこちらからてきぱきと能動的に処理をするわけですが　イベント駆動型の場合　むこうからしてくれと頼まれたことだけをするという受動的な処理をするわけです。

注2

X WINDOWで有名なものに似ているけど字体が違います。

注3

X-WINDOWのデモでよくあるあれです。

注4

現在DoGAでサポートできるネットは、

J&P HOTLINE

ASCII NET PCS

PC-VAN

NIFTY Serve

SPSネット

の5つです。

＊　＊　＊

今月のアップデータ

Gate open, Gate open, fire!"

なつかしのウルトラホーク1号です。ちゃんと3機に分離します。制作は　CADを担わせたら日本一（？）チームTOSAKAです　ただ　データが大きすぎるので　3機いっぺんにCADで読み込もうとすると、バスエラーが出るという　とんでもない作品です。

寺田の教育的指導

今月は私が「最遅投稿」です（DoGA関係でいちばん原稿が遅れている者をこう呼ぶ）　だから　あわてて文章を書いてます　文章も変になりそうです　締め切り直前の真っ最中ということで許してやってください

以前このコーナーでご紹介した堀山知己さん（あの本誌の表紙の方です　忘れた人はバックナンバーをひっくり返してみてください）から再びディスクが届きました　今度はなんとディスク5枚組です　この量だけでも期待が高まりますね　さあて、さっそく作品を見てみましょう

もし　こ　これは　　あの「人体モデル」が帰ってきた！（んな大袈裟な）あのままで終わるはずがないと思っていましたが　やはり作品としてまとめる方向で制作されていたようです。

付属の手紙によると（いつも丁寧なお手紙ありがとうございます）タイトルは「SWORD」、テーマは「時の流れと人」だそうです　うーん　なかなか難しそうですね。ストーリーは　サンプル版ということでよくわからないのですが「人」が家族（？）の「犬」（フレンダーそっくり）とともに「ドラゴン」と闘うというものらしいです

以前も書きましたように　作品に技術的な問題点はとくにありませんし　手紙に「動きなどのノウハウをお教えください」ともありますので　そのあたりについて書いてみたいと思います　とはいっても私も映像のプロというわけでもありませんので　DoGAのほかのスタッフの意見も参考にして　気づいた点を挙げてみます

まず　犬がドラゴンに飛びかかるシーンですが　少しカット割りが細かすぎる気がします　なかなか凝ったシーンで　犬が地面を蹴ってドラゴンに飛びかかるまでの間に　違う方向から見たカットが入るのですが　その間隔が短いので「目がちらちらする」ような感じを受けます　もう少し1つひとつのカットを長くしたほうがよいのではないでしょうか。

それと　飛びかかるときに犬の首がドラゴンとは別のほうを向いていますが　これは不自然だと思います。犬の闘志を表現するためにも　首はまっすぐドラゴンを睨みつけていた方がよいでしょう。フレームファイルで「vec」を使えばうまくいくと思います（首の向きを犬からドラゴンへ向かうベクトルで与える）。

また　ドラゴンに跳ねとばされた犬に人が走り寄るシーンで　倒れた犬を人が見つめているカットがありますが　このカットから次のカットへの切り替えが少し早いと思います。その次のカットで人が怒りのポーズをとっていることから考えて　このカットでは犬が殺された（あるいは重傷をした？）ことに対する「人」の悲しみ　悲しさが表現されているものだと考えられます。だとすれば、一般的にいって　もう少し長いカットでないと観ていて感情移入がしにくいのではないでしょうか

以上、かなり偉そうな書き方になってしまいましたが　一応常識的な面で判断して書いてみました。参考になれば幸いです　まだサンプル版ということなので　完成版に期待しています。頑張ってください。

ところで「MASAYA」シリーズはいまのところボツということですが　ちょっと残念です（個人的にはこちらのほうが好きだったのです）。あのメリーゴーラウンドも気に入ってたんですが　　暇があったらまた続きを作ってください。楽しみにしています。

それではまた次回に。しゃあ！

清水和人流
プログラミング道場
〔その3〕

段階的に鍛えていくべし

Shimizu Kazuto

みなさんお待たせいたしました。清水和人流プログラミング道場の3回目の講義です。今回はスプライトを使ったテーブルテニスゲームを作りながら、プログラミング発生法というものを理解していただくのが狙いです。

さて、今日も1日お疲れさま。デモ、寝る前にちょっと愛機を立ち上げる。その姿勢が未来の大物を育てるのである。なにせBASICで遊べる程度のプログラム力があれば、将来さまざまなソフト関係の仕事をこなせるからである。偉大な教育の場、それこそがBASICの世界なのである。

逆に見るとこんなふうにも考えられる。BASICで遊べる程度の腕があれば、多少人格的にはフラフラしていてもなんとか食える世の中なのである。個人技という言葉が聞かれるようになってから、社会人としても立派な人は日本には少なくなってしまったのである。だからといって学校の勉強をなまけてみたりすると、これがまた食えなくなるから、ヤル気はなくさないように。

まあ、とにかく寝る前に愛機を立ち上げたということで、私は今日も今日とてその前に座ったのである。何をしようという気持ちもまだない。いわば無の状態である。それでは、この立ち上げが無意味かというとそうでもない。それは心の中に「何かやろうかな」という小さな火を灯す。この小さなきっかけが大切なのである。これで次第に好きになって、病みつきへの道は遠くない。だいたい、BASICで挫折する人は個人的な計画を立てすぎるのである。凄いプログラムを組もうと思って、あれもこれも思いつく機能はすべて取り込む、なんてやっているとバグのオバケになってしまい「もう二度とBASICなんて」と別離を決心するハメになるのだ。そりゃ、X68000は立ち上げに時間がかかるし、立ち上げてもTVも見られない。しかし、夜にプロ野球ニュースを見るより、もう少しはマシなのではないだろうか。

1.今日はスプライトから

X68000を買った人には、カタログや広告のさまざまな宣伝文句につられて買った人も多いだろう。でも、その機能を自分で使っている人は案外少ないそうである。フルカラーグラフィック、FM音源、ADPCM、スプライト機能……etc. そういえばどれも昔のパソコンにはない、新しい用語（もうかなり古いが）ばかりである。

これらの機能は制限つきでBASICから使用が可能である。さて今日は何をやろうかな。んー、スプライト。これなら簡単そうである。スプライトといっても飲み物ではない（あー、言ってしまった）。RPG（ロールプレイングゲームだよ）に出てくる例の妖精ってやつである。お花畑をちらちら飛び回る。これがスプライトである。画面内を自由に動かせるキャラクタ、これがX68000スプライトである。

スプライトの使い方を1つひとつ説明するのは、ここでの目的ではない。それはBASICマニュアルに任せるとして、まずはサンプルプログラムから入ろう。リスト1はBASICマニュアルのパクリである。実に味わいのないサンプルであるが、一応スプライトのなんたるかをすべて含んでいる。スプライトをどう使うかは、このリストとマニュアルの各命令を見比べて、何をやっているか見ればいい。このくらいの努力を惜しんでいては将来は閉ざされたといってよかろう。

これを走らせてみると、「こらまったどうしたか!」遅いのである。「なんやスプライトちゅうても、こんな遅いのかいな」と関西オヤジの登場である。まあBASICだということもあるが、そのほかにも2つ遅い理由がある。

1) 座標を1ドットずつ動かしている
2) SIN関数の計算が遅い

1)については、動かすのに2ドットごとにすれば2倍の速さになるし、2)についても、SIN関数なんか使ってまでスプライトさせなくたっていいのである。とにかくドキッとさせられるサンプルだ。

そこで、もっと単純なスプライトを見るため、とりあえずリスト2を作ってみた。

●リスト1

```
10 int a1,x,y
20 dim char SP(1,255)={
30   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
40   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 2, 2,
50   0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 2, 2, 2, 3, 3,
60   0, 0, 0, 0, 0, 0, 0, 1, 1, 2, 2, 3, 3, 3, 1, 1,
70   0, 0, 0, 0, 0, 0, 1, 1, 2, 2, 3, 3, 4, 4, 1, 0,
80   0, 0, 0, 0, 0, 1, 1, 2, 2, 2, 1, 4, 4, 0, 0, 0,
90   0, 0, 0, 0, 1, 2, 1, 1, 1, 4, 4, 0, 0, 0, 7, 7,
100  0, 0, 0, 1, 2, 1, 1, 1, 0, 0, 0, 7, 7, 7, 0, 0,
110  0, 0, 0, 1, 2, 1, 1, 4, 0, 7, 7, 4, 0, 0, 9, 9,
120  0, 0, 1, 2, 3, 1, 0, 0, 7, 0, 0, 0, 1, 1,10,10,
130  0, 0, 1, 1, 3, 1, 1, 0, 7, 4, 0,10,10,10,11,11,
140  0, 1, 1, 3, 0, 4, 0, 7, 0, 0,10,10,11,11,12,12,
150  0, 1, 2, 3, 4, 5, 0, 7, 0, 0,10,11,12,12,12,12,
160  0, 1, 2, 2, 4, 0, 0, 7, 4, 0,10,11,12,13,11,11,
170  1, 2, 3, 1, 5, 4, 7, 0, 0,10,11,12,13,11,11,11,
180  1, 3, 3, 1, 1, 0, 7, 0, 0,10,11,12,13,11,15,14}
190  screen 1,3,1,1
200 sp_clr(0,255)
210 sp_off(0,127)
220 sp_disp(1)
230 sp_def(0,SP)
240 for a1=1 to 15
250   sp_color(a1,rgb(a1*2,a1*2,a1*2))
260 next
270 for a1=0 to 360
280   x=a1*2-10
290   y=sin(2*a1*3.1416/180)*100
300   sp_move(0,x,100-y,0)
310 next
320 goto 270
```

これはアクビが出るほど簡単だが、画面の中を円角が飛び回るのである。これならどうやらスピードも許せるようだ。ちなみに、220、230行のdx、dyの値を大きくすればスピードは速くなるし、小さくすれば遅くなる。

このプログラムはBREAKしないかぎり、ずっと動いているので、動かしたままにして冷たいものでも飲もう。最近のお勧めは見えないゼリードリンクのアップルである。ちらっと沈黙の艦隊に目を通すのもよかろう。で、目を上げるとまだスプライトしている。ずっと見ているうちに、「これゲームにならないかな?」と考える。

あっ!

そう、まさにこれと同じような経過をたどってゲームのヒントを思いついた人が、なんと20年前、江戸時代の末期にいたのである。そしてそれがTVゲームの元祖となった「テーブルテニス」である。

いまでこそファミコン、ゲームボーイを筆頭とするコンピュータゲームが全盛であるが、その先駆けとなったゲームマシンがあったのである。それは家庭のテレビにつなぐとテーブルテニスが楽しめるという、いまから考えるととんでもなく低レベルなものだが、当時若干売れた（現に私の友達も買った）。

リスト2はまさにこのゲームの基本なのである。これの左右にパドルが現れるともうそれだけでゲームなのだ。

ここまでくるとパドルを動かさなくてはならない。しかも、2人用ときて、また、

あっ!

●リスト2

```
10 int x,y,dx,dy
20 dim char SP(255)
30 /* initialize
40 screen 1,1,1,1
50 /* sprite pattern definition
60 for i=0 to 255:SP(i)=1:next
70 sp_clr(0,255)
80 sp_off(0,127)
90 sp_disp(1)
100 sp_def(0,SP)
110 sp_def(1,SP)
120 sp_color(1,rgb(30,30,30))
130 /* draw wall
140 col=rgb(30,30,30)
150 box(0,0,511,10,col,&HFFFF)
160 paint(250,5,col)
170 box(0,500,511,511,col,&HFFFF)
180 paint(250,505,col)
190 /* initial position & velocity
200 x=200
210 y=200
220 dx=4
230 dy=4
240 if y<=10 then dy=-dy
250 if y>=500 then dy=-dy
260 if x<=10 then dx=-dx
270 if x>=500 then dx=-dx
280 x=x+dx
290 y=y+dy
300 sp_move(0,x,y,0)
310 goto 210
```

である。X68000のBASICでは同時に2つのキーセットができないのである。つまり2人の入力を同時に判断できないのだ。この逃げ道として最も安易な方法が「ジョイスティックを使う」である。X68000のユーザー[illegible]ジョイスティックを1つくらいもっていそうである。2つとなると厳しそうなので、一方はキーボードだ。こうして出来上がったのがリスト3である。

リスト3はまさに「テーブルテニス」の

●リスト3

```
100 int x1,x,y,dx,dy,i,k$,ip1,ip2,y1,y2,[illegible]
200 str ks,k$
300 dim char SP(255)
400 /* initialize
500 screen 1,3,1,1
600 /* sprite pattern definition
700 for i=0 to 255:SP(i)=1:next
800 sp_clr(0,255)
900 sp_off(0,127)
1000 sp_disp(1)
1100 for i=0 to 10:sp_def(i,SP):next
1200 sp_color(1,rgb(30,30,30))
1300 /* draw wall
1400 col=rgb(30,30,30)
1500 box(0,0,511,10,col,&HFFFF)
1600 paint(250,5,col)
1700 box(0,500,511,511,col,&HFFFF)
1800 paint(250,505,col)
1900 repeat
2000 ip1=0:ip2=0
2100 randomize
2200 locate 21,1:print ip1:locate 41,1:print "-":print ip2
2300 /* initial position & velocity
2400 if x1=0 then dx=21 else dx=-21
2500 x=255:y=210:dy=11
2600 x1=1:y1=211:dx1=0:dy1=21
2700 x2=510:y2=211:dx2=0:dy2=11
2800 /* start
2900 repeat
3000 /* boundary condition
3100 if y<=10 then dy=-dy
3200 if y>=500 then dy=-dy
3300 /*if x<10 then dx=-dx goto 6000
3400 if x<=10 then {
3500   if y>=y1-21 and y<y1 then dx=18:dy=-12
3600   if y>=y1 and y<y1+10 then dx=abs(dx):dy=-dy
3700   if y>=y1+10 and y<y1+21 then dx=abs(dx):dy=-11
3800   if y>=y1+12 and y<y1+80 then dx=abs(dx):dy=0
3900   if y>=y1+18 and y<y1+81 then dx=abs(dx):dy=14
4000   if y>=y1+80 and y<y1+90 then dx=abs(dx):dy=21
4100   if y>=y1+90 and y<y1+100 then dx=18:dy=12
4200 }
4300 /*if x>=500 then dx=-dx goto 6000
4400 if x>=490 then {
4500   if y>=y2-20 and y<y2 then dx=-18:dy=-12
4600   if y>=y2 and y<y2+10 then dx=-abs(dx):dy=-21
4700   if y>=y2+11 and y<y2+12 then dx=-abs(dx):dy=-10
4800   if y>=y2+12 and y<y2+80 then dx=-abs(dx):dy=0
4900   if y>=y2+18 and y<y2+81 then dx=-abs(dx):dy=10
5000   if y>=y2+18 and y<y2+80 then dx=-abs(dx):dy=01
5100   if y>=y2+20 and y<y2+101 then dx=-18:dy=32
5200 }
5300 /*key or joystick sense
5400 ks=inkey$(0)
5500 /*stick(1)
5600 k2=stick(1)
5700 if ks="8" and y1>10 then y1=y1-dy1
5800 if ks="2" and y1<410 then y1=y1+dy1
5900 if k2=8 and y2>10 then y2=y2-dy2
6000 if k2=2 and y2<410 then y2=y2+dy2
6100 /*movement of ball and pads
6200 x=x+dx
6300 y=y+dy
6400 for i=1 to 5:sp_move(i,0,y1-10+10*i,1):next
6500 for i=1 to 5:sp_move(i+5,502,y2-10+10*i,1):next
6600 if x>=0 and y>=0 then sp_move(0,x,y,0)
6700 until x<0 or x>510
6800 /*point increment and display
6900 if x<0 then sp_move(0,0,y,0):ip2=ip2+1
7000 if x>500 then sp_move(0,500,y,0):ip1=ip1+1
7100 locate 10,1:print ip1:locate 11,1:print "-":print ip2
7200 repeat
7300 locate 20,00:print "[illegible]":ks=inkey$
7400 until ks="s"
7500 locate 21,10:print "[illegible]"
7600 until ip1=pmax or ip2=pmax
7700 /*display winner
7800 locate 20,1
7900 if ip1>ip2 then print "won by player 1 ";ip1;"-";ip2;" !"
8000 if ip1<ip2 then print "won by player 2 ";ip1;"-";ip2;" !"
8100 repeat
8200 locate 21,20:print "[illegible] ":ks=inkey$
8300 until ks="s" or ks="e"
8400 locate 22,20:print
8500 until ks="e"
8600 screen 1,0,1,1
8700 end
```

最も簡単なバージョンでもある。逆にいうとあまりにも味気ないバージョンでもある。でもまあこんなリストでもいろいろ苦労はある。

1) IF文が増えたためにスピードが落ちた。このため、何ドットも飛ばして動かすことでスピードアップさせざるを得ない(BAStoCコンパイルの助けは借りないのだ)

2) パドルと玉の速度差が微妙。とんでもなく難しくなったり、簡単になったりする

3) 変化を持たせるために、パドルの当たる位置によって跳ね返る方向を変えた

こんなものでも結構1時間以上かかるものである。もう寝よう。1日のうちにあまり深入りすると、アキがくる。自分をアキさせないことが大切なのである。寝ることによってまたよいアイデアが湧くかもしれない。

2 改造はアイデア次第

さて、自分で作ったゲームのよさはいくらでも改造できることにある。市販のソフトはいくらよくできていても、この改造に関してはほとんど無力である。ゲームは楽しめても自分の想像力を生かすところがない。自作なら、たいしたことのないゲームでも遊べると同時に自分の想像力を生かし、またプログラム力をアップすることができる。この差は歴然としている。

そして自作用プログラム言語も適した言語がBASICなのである。自作用言語、それがBASICであり、しかもX68000にはBASICからC言語への変換まで用意されている。そのため、多少普通のBASICと異なるが、なあに、Cのインタプリタだと思えば、こりゃええと思えてくる。そこまでめぐまれたX68000。しかもおおかたの機能はBASICでサポートされているとくれば、もうゲーム機だけにしておくのはナンセンスである。

さて、昨日作ったこの単純版「テーブルテニス」どんなふうに改造してやろうかと15分考える。「これは遊いけど難しそう」「これは簡単に変えられるけどいまいち」といったアイデアが次々と湧いてくる。これがプログラム後生説である。

プログラムの機能や使用をあらかじめ詳細に決め、設計を整えてからコーディングにかかる。このやり方を前生説とすれば、簡単な雛形をまず作っておき、それに次々と機能を加えていって大きくふくらませていくやり方が後生説である。このやり方はソフトハウスの人にはお勧めできないが、一般パソコンプログラマには最適な方法である。少しずつ作るのでその都度試すことができ、デバッグは簡単になるし、思いついたことをどんどん形にしていける。

今回のテーマはスプライトだが、第2のテーマはバージョンアップ、すなわち改造をしていくことである。

1) 改造その1 パドルの前後の動き

まず最初に思いついたのが「パドルを前後に動かせないか」ということである。テニスでいえば「前に出る」。これができるとゲーム内容もぐっと深まるのではないか。

この改造はパドルを動かすこと自体は簡単である。Y座標だけでなくX座標も変えてやればよいのである。それをテンキーやジョイスティックの前後方向に対応させればよい。ついでに斜めにも動かせるようにした。

気をつけるのは前後の境界処理である。ある場所以上には進めないように、X座標で判断する必要がある。それと、玉がパドルに当たったかどうかの判断が大きく変わる。いままでは両方の境界でのみ判断すればよかったが、この改造でパドルのX座標に合わせて判断することが必要になった。

そんなことを考えつつ、改造した修正部分がリスト4である。これをリスト3の上に打ち込めば改造完了である。元の行を変えたところや、新たに挿入したところもある。そのためにリスト3では行番号を100おきにリナンバし、改造しやすくしておいたのである。

この改造部分において2600から2700行のdx1、dx2を変えることによって、前後の動きの飛び幅を変えられる。極端にdx1=dx2=200とすれば前後2カ所をワープするだけになり、スムーズな移動とはまた違った楽しみ方ができる

2) 改造その2 カベを作る

このタイプのゲームにつきものなのが、玉を跳ね返す「カベ」である。これをいろんなバリエーションで置くことによって玉の行方がわからなくなり、ゲームはさらに複雑になる。といっても「カベ」だらけにしてしまうと難しすぎて面白くなくなってしまう。そのへんの呼吸が大切である。

カベをある一定の形に置いてしまうと、何回かやると飽きてしまう。そこで、1ゲームごとに乱数を用いてカベを出現させ、バリエーションを楽しむことにしよう。

リスト5がこのための改造部分である。リスト3にこのリストを重ねるとカベのあるゲームが楽しめる。

ここでのポイントはやはり玉がカベにぶつかる処理である。その位置にカベがあるかどうかは座標を比べるしかないので、少々面倒である。もう少し便利なコマンドがほしいところだ。

もうひとつのポイントはどのくらいカベを作るかである。ここでは画面の中央部、2つのパドルの中間のみに出現させた。それもあまりどぎついものでなく、ちょっとだけ出したほうが、ゲームの本質を保てていいようだ。

プレイ中に現れたり、動くカベというのも考えられるが、ここは深入りせずに別の機能を考えよう。

3) 改造その3 玉を2つに

さて、このゲームを一気に難しくしてしまおう。玉を増やすのである。これの基本的なプログラムはいままでのアルゴリズムと一緒である。ただループ1回につき2つの玉を別々に動かせばいいのである。意外に簡単な改造ですむ(リスト6)。

●リスト4

```
3410 |
4400 if x>=n1-10 and nt<=n1+10 then |
4410 |
440d if n1>n2 and nt<=n2+32 then |
572d if (in1d="7" or in1d="8") and y1t>=1d then y1=y1-dy1
d740 if (in1d="1" or in14="4" or in14="7") and (1t-10
     then n1=n1-dx1
0d00 if in2 = d  and y2t-td  then y2=y2-dy2
d020 if in2 = 7  or in2 = d ) and y2t 10 then y2=y2-dy2
0040 if (in2   1  or in2 = 4  or in2 = 7 ) and (2t<=400
     then n1=n1 dn1
0020 if (in10="1" or in10="3") and y1<-410 then y1=y1+dy1
d040 if (int0="3" or in10="d" or in1m="0") and n1<=200
     then x1=x1+dx1
0020 if (in2 = 1  or in2 = 4 ) and y2<=410 then y2=y2+dy2
004d if (in2 = 2  or in2 = 0  or in2 = 9 ) and (2t<=20d
     then x2=x2+dx2
```

●リスト5

```
100 int y3n,y3o
2040 line(2dd,10,200,405,0)
20d0 y3n=rand() mod 200 y3n-y2n+(rand() mod 100)+100
200d line(2d0,y3n,2d0,y3n,0dd30)
42d0 if x>=23d and n<=275 and y>y3n and y<y3o then dx=-dx
```


ただこれをプレイするのは大変である。いい勝負をするには腕のたつ2人のプレイヤーが必要である。まあ仲のいい兄弟くらいであろう。

だいたいこの手のゲーム、最初は必ず2人のうちどちらかが勝ち続けることになっている。それは1人だけが熟練しているからである。2人で仲よくプレイするためには、できるだけ簡単なモードから慣らしていくのがよい。これが「お友達の法則」である。

話はそれるが、2人でとことんまで闘ったゲームというのは案外思い出に残るものである。私の場合ずいぶん古い話だが「ハイパーオリンピック」は実によく覚えている。もうひとりは別に知り合いではなかったのだが、同じゲームセンターでレベルの高いスコアを出しているうちに、一緒の台でプレイするようになったのである。お互い名前も知らない仲だが、結構楽しんだ。あのゲームはいかに速くボタンを押しつづけるかが勝負なのだが、2人のやり方はまったく違っていてその優劣を競ったのである。ゲームはライバルが大切である。

4) 改造その4 パドルが小さい

さて、次なる改造はさらにこのゲームを難しくしてくれる。パドルを小さくするのだ。いままで大雑把な感覚でやっていたのだが、より繊密な動きを要求される。昔懐かしいブロック崩しを思い出すではないか。みんなこれで腕を上げていったものである。

これも改造は簡単である。リスト7が改造部分である。いままでパドルに5つ分のスプライトを用いていた処理を変数NPに変えたのである。これによって玉の衝突判定も必然的に範囲が変わる。

小さいパドルでのプレイは非常な集中力を必要とする。玉の行方を必死で追わなくてはならない。細かいキー操作も必要である。このような状況ではキーボード入力よりもジョイスティックのほうがいいかもしれない。キーボードのリピート機能はあまりにも貧弱だから、きっと操作ミスをしてしまうだろう。X BASICの課題である。

さて、今日はいろいろな改造を試みたが意外に変更点が少なかったことはリストを見てもらえばわかるであろう。このように、あとから機能を追加していくやり方は、非常に楽なのである。それにはまず基本形として簡単なプログラムを組み、あとはアイデアを凝らす→追加、修正する、といった手順である。毎日少しずつこれをやっているうちに、すごいゲームができるかもしれない。

5) 改造その5 スピードの変化

しかし、このゲームを作ってみると、一生懸命やっても、いろいろと「大したない」と案外スピードが遅いことがわかった。とはいうものの、いまは使うことを拒否しているが、バックにはまだコンパイラという強い味方がついている。このゲームもコンパイルしたらもっとゆっくりする方向で修正をしないと、とてもやってられないスピードになるかもしれない。

さてしかし、インタプリタのままで、このうえさらに玉のスピードに変化をつけてみたくなった。ただ速くしたり遅くしたりするだけなら、変数dxやdyなどの値を大きくしたり小さくしたりしてやればよい。しかし、ここではプレイの最中に玉のスピードが変化するようにした(リスト8)。

このプログラム、dxやdyをむやみに大きくしたり小さくしたりすると衝突判定が難しくなってしまう。なぜ座標がお互い飛

●リスト6

```
12d int n,b,da,db
1160 sp_def(1,SP1)
24dd dn=-dn
2850 n=2dd  b=214        bb=10
315n if b<=10 then db=-db
3240 if b>=400 then db=-db
32d1 /*if nt=10 then dn=-dn goto 32d4
32d2 if nt=10 then |
3253 |
3204 if n)=n1-10 and nt<=n1+10 then |
3240    if b> y1-24 and b<y1         then da=1d  db<-32
324d    if b>=y1    and b<y1+10      then da=nbn(da) db<=24
3247    if b>=y1+16 and b<y1+42      then da=nbn(da) db=-10
324d    if b>=y1+42 and b<y1+4d      then da=nbn(da1) db=0
34d5    if b>=y1+40 and b<y1+04      then dn=nbn(da) db=10
32d0    if b>=y1+04 and b<y1+00      then dn=nbn(da) db=24
4201    if b>=y1+00 and b<y1+t04     then da=10  db=32
3292 |
3282 if n>=244 and n<=274 and b>=ydn and b<=y3o then da=-da
2254 if n>=420 then dn=-dn goto 3300
4260 if nt-400 then |
2200 |
3207 if n) x2 and nt<=n2+42 then |
320d    if b> y2-24 and b<y2         then dn=-16 db= 34
420d    if b>=y2    and b<y2+10      then dn=-nbn(da) db=-24
3270    if b>=y2+10 and b<y2+02      then dn=-nbn(dn) db=-15
3271    if b>=y2+22 and b<y2+4d      then dn=-nbn(da) db=0
3272    if b>=y2+40 and b<y2+04      then da=-nbn(dn) db=10
2274    if b> y2+d0 and b<y2+60      then dn=-nbn(dn) db=44
4274    if b>=y2+bd and b<y2+104     then dn=-16 db=32
3270 |
02d0 n=n+dn
0300 b=b+db
0040 if nt 0 and bt.0 then sp_move(1,n,b,1)
0700 nmt(1) ntd or x<4d0 or nth or n 4bb
0000 if nt0   then sp_move(1,  ,1)    (p2+(p2+1
7050 if nt450 then sp_move(1, 400, ,1)  (p1=(p1+1
```

●リスト7

```
170 int np=3
32d0    if b>=y1              and b<y1+10*np/0    then da=nbn(dn) db=-24
3257    if b>=y1+16*np/5 and b<y1+32*np/4    then dn=nbn(dn) db=-1d
4200    if b>=y1+32*np/5 and b<y1+48*np/d    then da=nbn(dn) db=d
32d0    if b>=y1+40*np/5 and b<y1+04*np/4    then da=nbn(da) db 10
3200    if b>=y1+64*np/5 and b<y1+80*np/4    then dn=nbn(da) db 24
3201    if b>=y1+0n*np/n and b<y1+0n*np/d+24 then dn=10  db=32
3204    if b>=y2         and b<y2+16*np/4    then dn=-nbn(dn) db=-24
3270    if b>=y2+10*np/0 and b<y2+32*np/5    then dn=-nbn(da) db=-10
3271    if b>=y2+22*np/5 and b<y2+4d*np/6    then dn=-nbn(da) db=0
3272    if b>=y2+40*np/0 and b<y2+0<*np/6    then dn=-nbn(da) db=1d
3273    if b>=y2+d<*np/d and b<y2+00*np/d    then dn=-nbn(da) db=24
3274    if b>=y2+00*np/0 and b<y4+00*np/d+24 then da=-16 db=42
3000    if y>=y1         and y<y1+1d*np/5    then dn=nbn(dn) dy=-24
4700    if y>=y1+10*np/0 and y<y1+22*np/0    then dn=nbn(dx) dy=-1d
3000    if y>=y1+32*np/0 and y<y1+40*np/0    then dn=nbn(dn) dy=0
4500    if y>=y1+40*np/d and y<y1+d4*np/0    then dx=nbn(dn) dy=10
4000    if y>=y1+d4*np/d and y<y1+0n*np/0    then dn=nbn(dx) dy=24
4100    if y>=y1+00*np/4 and y<y1+d0*np/0+24 then dx=1d dy=32
4000    if y>=y2         and y<y2+10*np/0    then dn=-nbn(dn) dy=-24
4700    if y>=y2+10*np/4 and y<y4+22*np/0    then dn=-nbn(dx) dy=-10
4000    if y>=y2+32*np/d and y<y2+40*np/0    then dn=-nbn(dx) dy=0
4500    if y>=y2+40*np/0 and y<y2+54*np/0    then dn=-nbn(dn) dy=10
0000    if y>=y2+d4*np/5 and y<y2+80*np/0    then dn=-nbn(dn) dy 24
h100    if y>=y2+d0*np/0 and y<y2+8d*np/0+24 then dn= 10 dy=32
d004 if in1m="2" and y1<-400-np*10 then y1=y1+dy1
0000 if in2 = 2  and y2<-400-np*10 then y2=y2+dy2
6400 for i=1 to np:sp_move(i, n1,y1-10+10*i, i):next
0000 for i=1 to np:sp_move(i+d,n2,y2-10+10*i, i+d):next
```

び飛び」変わっていくので、その軌跡が変わってしまう。よくわからなくなってしまうのである。dx dyの値はこの点で微妙な注意が必要となる。こういう類の設定はあまり考えすぎると疲れてしまうので、ある程度見当をつけたら少しずつ値を変えて試してみることが必要である。こういうときは2人でプレイできないので、片方の壁へ全部を反射させてしまうのがよい。リスト8の3300行と4300行はコメントになっている。このデバッグ方法のための処理である。

6) 改造その6 変化球

その昔、星飛雄馬は球質の軽さから直球に不安を感じ、大投手金田に変化球を教えてくれと頼んだ。そこで金田は「若いうちからそんなことでどうする。大リーグもびっくりするような新しい球を考えるんや」と一喝したという。かくして大リーグボールは生まれてくるわけだが、それにあやかって変化球を取り入れてみようではないか。

変化球の処理は思ったほど難しくない。要は、何もないところで玉の速さdx、dyを別々に変えてやれば曲がるのである。問題は「連続してぐぐーっと曲げる」か「ある場所でカクッと曲げる」かの選択である。でも筆者はそこに壁があるのと似たようになってしまうので、ここでは連続して曲げていくことにしよう。それにはループが回るごとに、少しずつdxやdyを変化させてやればよいのだ（リスト9）。

これをプレイしてみると玉が逃げるように動いたり、逆に向かってきたりして、いままでとはまったく違った感じのゲームになる。曲がり方がまったく同じだと飽きてしまうので乱数を用いた。ゲームに乱数はつきものであるから使い方は覚えておいたほうがよい。この改造も玉とパドルの衝突判定がネックになってくる。

●リスト8

```
120 irt lc
2100 lr=0
2600 if lc mod 100=80 then dr=2*da da=2*da
2670 if lc mod 100=10 then da=dr/2 da=dr/2
3000 if r>=r1-rbs(da) and r<=r1+rbs(da) then (
3001 if r>=r2 and r<=r3+rbs(da)*3 then )
4000 if r>=r1-rbs(dx) and r<=r1+rbs(dr) then (
4300 if r>=r2 and r<=r3+rbs(dx)*0 then )
4400 lr=lr+1
4500 /*if (r1400 and dx>0) then dx=-dx
4600 /*if (r1400 and da>0) then da=-da
4640 /*if (r116 and dx<0) then dx=-dx
4665 /*if (r116 and da<0) then da=-da
4700 until (x16 and dx<0) or (x>496 and dr>0) or (a<0 and da<0)
 or (a>400 and da>0)
```

●リスト9

```
2900 if lc mod 100>=80 and lc mod 100<=80 then dy=dy+0 db=db+0
3000 if lr mod 100>=80 and lr mod 100<r00 then dy=dy-2 db=db-2
```

●リスト10

```
510 dim char SP2(200)
600 rrarrlr ,0
700 for i=0 to 16 for j=0 to 15
760     SP1(i*16+j)=int(sqr((i-8)*(i-8)+(j-8)*(j-8)))/2+3
800     SP2(i*16+j)=int(sqr((i-8)*(i-8)+(j-8)*(j-8)))/3+0
810 next next
910 for i=1 to 10 sp_def(i,SP2) next
1000 for i=1 to 7 sp_color(i,rgb(30-i*1,0,0)) next
1100 for i=2 to 1 sp_color(i+6,rgb(0,20-i*3,0)) next
1200 (c)rgb(16,20,30)
1800 line(0,0)-(250,y3),40000)
```

●リスト11

```
110 int da=dx
200 m_lrlo(1) for i=1 to 6 m_alloc(i,200) m_assign(i,i) next
210 m_ork(1,"@1122 200rrbrgfugfedcegbcrgbcegbcrgb (")
211 r_orh(2,"@00r2r0 200o0 e1do e0dde0g0 e1dg e0f0r0d0 d10d f0r0
e0r0r0r1b0r0e0(0r0d0 (")
272 r_irh(3,"@1133(e(r(")
312 r_orh(1,"@10(01gbgbgbgb")
371 r_irh(0,"@10cr0r(0r0-e d)(d0 e10c0 c2b(dr1e2")
510 r_orh(0,"@10g2r0ed0g0e0d0e0c1e0g0r0e0e0e0d0r0r0e0d0c(0d(0r1")
600 m_plre(1)
700 r_rerp(1)
830 m_play(1)
4600 if y<=10 then dy=-dy m_plry(3)
4610 if b<=16 then db=-db m_play(3)
4800 if a>=100 then dy=-dy m_plry(2)
4810 da=-da dr=-dx
4216 if b>=100 then db=-db m_plry(2)
5000 if dr<>dar then m_play(4)
5001 if dar(da then m_play(4)
6010 m_rerp(2) m_plre(0)
7000 locate 20,20 print "
7010 m_plre(0)
8200 locate 20,1 print "
8300 locate 22,20 print "
```

3.改造のつぎは修飾

以上6つの改造を試みたが、プログラムは改造ばかりが能ではない。もうひとつの大切なものが仕上げのデコレーション、修飾である。今回は2通りの修飾を考える。

1) 修飾その1 色を使う

いままでのリストでは味気のない真四角の玉と長方形のパドルが動いているという単調な画面であった。X68000の65536色を生かすこともせず、使った色もたったの2色（白と黒）である。せっかくスプライトは色などに関係なくスピードを保てるのだから、そろそろ使ってみたい。

このための修正リストがリスト10である。この程度の変更で画面の感じもグッと違ってくる。残念ながらセンスのない私にはこの程度しかできないが、センスのある（二科展に入選するような）人がやればきっと魅力的な画面になるだろうと思う。

2) 修正その2 音をつける

なんといったって効果音のないゲームは寂しい。これはほかの機能をほとんど考えずにつけ加えることができるから楽である。効果音はFM音源やAD PCMを使えばよい。AD PCMはデータを作らなくてはならないので、ここではFM音源を使おう。

改造例がリスト11である。ちょっとスピードは気になるが、自分の好きな音楽を用いれば自然とよい気分でプレイができようというものである。BGMを流すのにもFM音源でという手もある。最も簡単で最も効果のある修飾である。

以上、すべての機能を統合してできたのが最後のリスト12である。今回のテーマは「スプライト」であろうか「操作説」であろうか。まあ、ゲームは簡単なものから始め、改造をし、最後に修飾すればそんなに苦労なく進んでいけるものなのである。

せっかくのX68000なのだからスプライトやFM音源やカラーを使って面白いプログラムに変えていこうではないか。

●リスト12

```
[illegible]
```

PurePASCAL

PASCALプログラミングへの招待(6)

実行時チェック・Cとのリンク

Fujiki Takeshi　Fujii Yoshimi
藤木健士/藤井義巳

PASCAL入門の連載もいよいよ大詰め。今回は"高級言語"PASCALの一面でもある実行時のエラーチェックとより高度な運用法のためのC言語とのリンク，コンパイルオプションなどについて解説します。

1 実行時チェック

実行時チェックというのは実際に処理を行ったときに発生するエラーのチェックのことです。

たとえば

var　i:1..10;

と宣言した場合，変数iは1より小さい値や10より大きな値をとることができません。このような変数の範囲を越えるといった種類のエラーは文法エラーと異なりコンパイル時には検出することができません。実際に処理を実行して，変数に値が代入される時点になってはじめてその値がわかるからです。では，どのようにしてこのチェックを行うのでしょうか？　それは，コンパイルするときに実行時エラーが発生する可能性のある部分にはチェックルーチンを組み込んでしまうのです。

それでは例をとって説明することにしましょう。リスト1を見てください。このプログラムはいくつかの数字を入力してその数字の中の最大のものを表示するプログラムです。このプログラムでは最大5個の数字から最大値を求めることができます。これをコンパイルして実行すると入力要求を出してくるので数字を次のように入れてみましょう。

123 53 36 652 0

すると次のように最大値を返してくるでしょう。

最大の数値　　652

それでは今度は入力要求のあとに次のように入力してみましょう。

431 1431 23 43 78 867 234 0

すると今度は実行時エラーが発生して次のように表示されるでしょう。

実行時エラー　　LINE 18
スカラーまたは部分範囲が範囲を越えています。

これは18行目のreadln(a[i])を実行しようとした際にiが6を越えてしまったために起こったのです。これは配列aは整数6つ分の領域しか宣言していないためにこのような結果になるのです。

もしこの確認を行わずに実行を続けてしまうとa[7]に数値を入力してしまうことになり，確保していない記憶領域の更新をしてしまいます。この実行時チェックの機能がなければ，別の変数の値がいつのまにか変わっていたり，悪いとき（良いとき？）にはアドレスエラーなどでプログラムが止まってしまうこともあるでしょう。この手のバグを探すのは非常に大変です（経験したことはありませんか？）。

実行時に発生するエラーはほかに，0で割ろうとしたり，オーバーフローが発生したり，メモリが足りなくなったりしたときにも発生します。

一般にPASCALで書かれたプログラムはこのように実行時のエラーの検出を行います。Cでは実行時のエラーのチェックはやりませんので，そのようなチェックを行いたいときにはプログラマがチェックの部分をコーディングする必要があります。しかし，PASCALでは実行時エラーが発生する可能性があるすべての部分で実行時チェックの処理を行うので処理する時間がCで書かれたプログラムよりも長くなり，実行ファイルのサイズもチェックルーチンの分だけ大きくなります。

リスト1

```
 1	program MaxNumber(input,output);
 2
 3	const	TableSize = 6
 4
 5	var	i:integer;
 6		a:array[1..TableSize] of 0..MAXINT;
 7		max:integer;
 8
 9	begin
10
11		for i:=1 to TableSize do
12			a[i] := 0;
13
14		i:=0;
15		writeln('数字を入力して下さい。最後の数字の後は
0を入力してください');
16		repeat
17			i := i + 1;
18			readln(a[i])
19		until a[i] = 0;
20
21		max := 0;
22		for i := 1 to TableSize do
23			if max < a[i] then
24				max := a[i];
25
26		writeln('最大の数値',max)
27	end.
```

2 Cとのリンク

前回はPASCALのプログラム中にアセンブリ言語で書いた処理を書いてそれを実行する方法を紹介しました。今月はそれに続いて今回はCのライブラリ関数を外部関数としてPASCALのプログラムから呼び出す方法を紹介します。リスト2を見てください。これはファイルを行単位でソートするプログラムです。29行から33行目までで外部関数の宣言を行っています。

外部関数の宣言はextern指令を使って次のように書きます。

```
FUNCTION　関数名(パラメータリスト)：戻り値；
extern;
PROCEDURE　手続き名（パラメータリスト）；
extern;
```

リスト2のサンプルプログラムではファイル入出力関係のCのライブラリ関数を宣言しています。Cでは大文字と小文字の区別をするのでPurePASCALでも外部関数を宣言するときには大文字と小文字を区別するようにしています。注意してください。ここでもうひとつ注意すべきことがあります。30行目を見てください。

```
function　fgets(fp:FILEP; n:integer; var s:
STR255):STRP;extern;
```

となっていますね。ところがCのライブラリマニュアルを見るとfgetsの引数について次のように書いてあります。

```
char	*fgets(string, n, stream);
char	*string	/*　読み込んだ文字列の格
			納領域へのポインタ　*/
int	n		/*　読み込む文字数　*/
FILE	*stream	/*　FILE構造体へのポ
			インタ　*/
```

引数の順序が逆になっていますね。これはPASCALでは引数がCとは逆に積む（11月号を参考にしてください）からです。

それからファイル構造体へのポインタの型ですがFILEPつまり^CHARというようにPASCALのプログラム上では文字へのポインタというふうに記述していますが，これは4バイトの領域をとる型であればなんでもかまいません。ただfopenで得た値をほかのファイル入出力関数に渡したいだけなのでこれで十分なのです。

もしPASCALのプログラムの中でファイル構造体の内容を参照したいなら，

FILEP = ^CHAR

の代わりに次のように書けば一応アクセスできるのではないかと思われます。ただしPurePASCALでのアライメントがCと一致していないと思いますので，普通のやり方では完全に正しい値がくることを保証できません。使う必要のある場合はいろいろ試してみてデータが正しく受け渡されるかたちのレコード型を定義してください。

```
type
	FILESTRUCTURE = RECORD
		_ptr:^char;
		_cnt:integer;
		_base:^char;
		_flag:integer;
		_bsize:integer;
		_file:char;
		_pback:char;
	end;
	FILEP = ^FILESTRUCTURE;
```

これはCのインクルードファイルstdio.hに記述してある次の構造体と同じかたちです。

```
extern	FILE	{
	char	*_ptr;
	int	*_cnt;
	char	*_base;
	int	*_flag;
	int	*_bsize;
	char	*_file;
	char	*_pback;
	}	*_iob[_NFILE]
```

ここで

type　FILE = record

というようにFILEという名前を使わないでください。FILEという名前は事前に宣言された型としてすでに存在します。それでここで上のように書いてしまうとreadやwriteでFILEに書き込むことができなくなってしまいます。注意してください。

あとは通常のPASCALのプログラムと同じです。それではコンパイルしてみましょう。いつものように次のようにコマンドを入力してください。

>pascal sort.pas

するとコンパイルとアセンブルを実行して終了します。外部関数を使っていないとき（extern指令を使っていないとき）はリンクまで実行しますが，外部関数を呼んでいるときにはコンパイラはどこに外部関数が存在するかわからないのでリンクは実行しません。それでマニュアルでリンクをする必要があります。今回呼んでいるファイル入出力関数はXCの場合CLIB.Aに入っていますの

で、次のようにコマンドを入力してください。

>LK sort o paslib a clib a

CLIB AやPASLIB Aがカレントディレクトリにない場合はフルパスで指定してください。以上でSORT Xという実行ファイルができあがります。それでは実行してみてください。

Input File name :

と入力ファイル名を聞いてきますのでファイル名を入力してください。そのあと次のように出力ファイル名を聞いてきます。

Output file name :

なにも入力せずにリターンキーを打つとコンソールに結果が出力されます。ファイルに出力したいときはファイル名を入力してリターンキーを押してください。ファイルの内容が並べ替えられているでしょう!

ちなみに、このプログラムではソートのアルゴリズムとしては総合的にもっとも速いといわれているクイックソートを使っています。簡単にアルゴリズムを説明すると、どれか要素をひとつを取ってきてそれ以外のものをそれより大きいグループと小さいグループに分けて、それの分けたグループの中でも同じ動作を行い、それより大きなグループと小さなグループに分ける。これを続けてグループの中が1になったらソートを終了する、こんな感じです。

このアルゴリズムは再帰を使って書くと簡単です。サンプルプログラムでもそのようになっています。興味のある人はプログラムを解析してみてください。ソートのアルゴリズムにはこのほかにもいろんな方法がありますが総合的にクイックソートより高速なものは考え出されていません。もしこれより速いアルゴリズムを考えたらすごいですよ! ひと儲けできるでしょう。

話がそれてしまいましたが、この外部関数をリンクする方法を使うと、Cで提供されている豊富な関数を用いることができます。これで、プログラムの幅が非常に広くなったのではないでしょうか?

3 PurePASCALのコンパイルオプション

PurePASCALはいくつかのコンパイルオプションを使用することができます。それではそれらについてひとつずつ説明していきます。

ファイルインクルード

これはCにおける#includeと同じ機能を提供するもので、指定されたファイルをこの指令が書かれた位置に挿入します。

書式

{$Ifilename}

例

{$Igraphic.inc}

このように書くとgraphic.incというファイルをこの指令が書かれた位置に展開します。

インラインアセンブラ

{$asm} から {$endasm} までのあいだにアセンブリ言語のプログラムを埋め込みます。IOCSコールやDOSを使いたいときなどに使ってください。

書式

{$asm}

　　アセンブリ言語のプログラム

{$endasm}

インラインアセンブラに関してはOh!Xの11月号に詳しく説明してあります。参考にしてください。

スタック領域のサイズの指定

実行ファイルのスタック領域を1Kバイト単位で指定します。省略値は64Kバイトです。プログラム中で大きな配列などを使用したり、かなり深い再帰を行うときには大きめに設定してください。これによりスタックオーバーフローを回避することができます。

書式

{$Ssize}

例

{$S256}

256Kバイトのスタック領域を確保します。

ヒープ領域のサイズの指定

実行ファイルのヒープ領域を1Kバイト単位で指定します。省略値は64Kバイトです。NEW手続きで確保する領域が大きくなる可能性があるときには大きな値を指定してください。

書式

{$Hsize}

例

{$H256}

256Kバイトのヒープ領域を確保します。

実行時チェックのコード生成スイッチ

今回説明したようにPASCALは実行時にいろいろなエラーチェックを行います。そのために実行速度がある程度低下しますし、プログラムサイズも大きくなります。{$nocheck}を指定すると、そこから先の部分では実行時チェックのコードが生成されません。{$check}を指定すると、そこから再び実行時チェックが再開されます。コンパイル開始時には {$check} の状態になっています。プログラム作成時には {$check} を行い、デバッグが済んだら {$nocheck} にしてスピードアップするとよいでしょう。

書式

{$check}

{$nocheck}

*　　*　　*

11月号で説明したアセンブリ言語とのリンク、そしてCとのリンクを使えばかなりのことができると思います。いろんなプログラムを作ってみてください。

リスト2

```
{* sort.pas *}
(*
    ファイルを行単位でソートする。
    ファイルの入出力にCの標準関数を使用する。
    ヒープを使うので、大きなファイルをソートする時は
    コンパイラ時でヒープサイズを大きくするとよい。
      Oct 1988 By Chas8 *
*)

program sort(input, output);
const
    MAXLINES = 10000;    (* 最大1万行まで *)
type
    (* テキストファイルの1行は文字数は255文字まで *)
    STR255 = packed array[1..255] of char;
    STRP = ^STR255;
    FILEP = CHAR;      (* コンパイラをだましている *)
    (* ソートのための配列 *)
    STRING_ARRAY = array[1..MAXLINES] of STRP;
var
    [illegible]
    TextLines: STRING_ARRAY;
    lines: integer;
(*
    リンクするCの関数のEXTERN宣言。
    関数名の大文字と小文字は区別される。
    戻り値が不要な関数はprocedureとして宣言すればよい。
*)
    function fopen(var ltype: STR255; var [illegible]: STR255): FILEP; extern;
    function fgets(fp: FILEP; n: integer; var s: STR255): STRP; extern;
    procedure fputs(fp: FILEP; var s: STR255); extern;
    procedure fclose(fp: FILEP); extern;
    procedure putchar(c: integer); extern;

(*
    ここから先はPASCALのプログラム
*)
    (* Cの形式の文字列を標準入力から入力する。 *)
    function getline(var line: STR255): integer;
    var
      i: integer;
      c: char;
    begin
      i := 1;
      repeat
        read(c);
        if not eoln then
          begin
            line[i] := c;
            i := i + 1
          end
      until eoln or (i >= 255);
      line[i] := chr(0);
      getline := i - 1;
      readln
    end;

    (* テキストファイルをまるごと読み込む *)
    function readfile(var fn: STR255): integer;
    var
      i, j: integer;
      s: STRP;
      ltype: STR255;
      fp: FILEP;
    begin
      ltype[1] := 'r'; ltype[2] := Chr(0);
      fp := fopen(ltype, fn);
      if fp = nil then
        begin
          writeln('File open error < ', fn, ' >');
          readfile := 0
        end
      else
        begin
          i := 1;
          repeat
            s := fgets(fp, 255, s);
            if s <> nil then
              begin
                new(TextLines[i]);
                for j := 1 to 255 do
                  TextLines[i]^[j] := s^[j];
                i := i + 1
              end;
          until (MAXLINES < i) or (s = nil);
          fclose(fp);
          readfile := i - 1
        end
    end;

    (* ファイルに出力 *)
    procedure writefile(var fn: STR255; lines: integer);
    var
      i: integer;
      ltype: STR255;
      fp: FILEP;
    begin
      ltype[1] := 'w'; ltype[2] := Chr(0);
      fp := fopen(ltype, fn);
      if fp = nil then
        writeln('File open error < ', fn, ' >')
      else
        begin
          for i := 1 to lines do
            begin
              fputs(fp, TextLines[i]^);
              dispose(TextLines[i])
            end;
          fclose(fp)
        end
    end;

    (* 画面出力し解放 *)
    procedure WriteToScreen(lines: integer);
    var
      i, j: integer;
    begin
      for i := 1 to lines do
        begin
          j := 1;
          while (TextLines[i]^[j] <> Chr(0)) and (j < 255) do
            begin
              putchar(Ord(TextLines[i]^[j]));
              j := j + 1
            end;
          dispose(TextLines[i])
        end
    end;

    (* クイックソート *)
    procedure qsort(l, r: integer);
    var
      i, m: integer;

      (* 行を交換する *)
      procedure swap(var a1, a2: STRP);
      var
        a: STRP;
      begin
        a := a1;
        a1 := a2;
        a2 := a
      end;

    begin { qsort }
      if l < r then
        begin
          m := l;
          for i := l + 1 to r do
            begin
              if TextLines[i]^ < TextLines[l]^ then
                begin
                  m := m + 1;
                  swap(TextLines[i], TextLines[m])
                end
            end;
          swap(TextLines[l], TextLines[m]);
          qsort(l, m - 1);
          qsort(m + 1, r)
        end { if }
    end; { qsort }

    (* メインルーチン *)
    begin { MAIN }
      write('Input file name : ');
      n := getline(ifn);
    (* 出力ファイル名でリターンだけを入力すると標準出力になる。 *)
      write('Output file name : ');
      n := getline(ofn);
      lines := readfile(ifn);
      if lines <> 0 then
        begin
          qsort(1, lines);
          if n = 0 then
            WriteToScreen(lines)
          else
            writefile(ofn, lines)
        end
    end.
```

カード型データベース(3)

Izumi Daisuke 泉 大介

X-BASICプログラミング調理実習もいよいよ最終回となりました。カード型データベースの最後にテキストファイルを扱うためのツールを加えておきましょう。また、付録ディスクにはVS2 Xに対応したバージョンが収録されています。

前回、前々回の2回を使って、カード型データベースを制作しました。使い心地はどうでしょうか。今回は調理実習の最後として、カード型データベースの能力を拡張する簡単なツールを作ることにしましょう。また、ディスクに収めたcbase.xは連載で発表したものに若干の手を加えてありますので、リファレンスマニュアルを用意しました。参照してください。

データの出入り口

データベースはそれ単体で完結してしまうものではありません。データの入り口はキーボードだけであってはいけないのです。最も身近なデータの入り口はワープロです。日記やちょっとしたデータの記録などにもワープロを使っている方は少なくないだろうと思います。またワープロは、ディスプレイ以外のデータの出口としても有用です。多くのデータの中からさまざまな条件で検索をかけ、見つけ出したデータを文書に取り込んで体裁よく仕上げるという作業を考えてみるとよくわかると思います。

前回までに作成したデータベースは、こういった他のアプリケーションとのデータのやり取りに関してはまったく考えられていません。そこで今回は、データベースファイルをテキストファイルとして出力する、テキストファイルをデータベースファイルとしてデータベースに取り込む2つのツールを作ってみることにしましょう。

●テキストデータの形式

データベース、表集計、ワープロは、通常それぞれ独自の形式でディスクにデータを保存しています。このため、データベースのファイルをワープロで読み込んだり、ワープロの文書を表集計ソフトで読み込んで使うことはできません。

そこで、アプリケーション間で共通のデータ形式が考えられました。ただし、独自の形式に比べてディスク入出力の効率は落ちますので、独自形式の他に共通形式もサポートするという形で実現されるのが普通です。

最も有名な共通形式はCSV形式と呼ばれる方法です。表集計ソフトでいうなら、表の横方向のデータをカンマで区切って並べ、縦方向の各行は改行で区切ろうというものです。カード型データベースでいうなら、データとデータをカンマで区切り、カードとカードは改行で区切ることになります。ただし、共通であるはずのCSV形式も厳密な定義があるわけではなく、文字列をダブルクォートでくくるものがあったり、カンマの代わりにタブでデータを区切るものがあったりとまちまちです。データ共通化の道は、まだその途に就いたばかりだといえるでしょう。

今回採用したテキストデータ形式も、X-BASICで簡単に扱え、さらに実用になるスピードが得られるものという条件を無視するわけにいかず、CSVやSYLKなどの既存の数種類の共通フォーマットとは異なったものになっています。まず、カード上の各々のデータは改行によって区切られます。これはfreads関数で読み出せるようにという配慮です。そして、それぞれのカードは改頁と改行によって区切られています。freads関数で読み出した文字列が改頁（chr$(12)）だった場合には、カードはそこで終わると判断できるわけです。このデータ形式は、標準ワープロであるWP.X、標準エディタであるED.Xでそのまま扱うことができます。

テキストデータの出力

まずはデータベースファイルをテキストファイルとして出力することから考えてみましょう。現在使用されている全カードを対象にしようかとも思ったのですが、cards配列に登録されているカードだけに止めることにしました。条件に合致したカードだけを出力してワープロなどで再加工できるように、という配慮からです。

プログラムはリスト1のtextOutput関数です。2つのパラメータを受け取っていますが、最初のnfpはテキストを出力するファイルのファイル番号、次のnはcards配列に入っているカードの総数です。テキストを出力するファイルのオープンは、あとで説明するtools関数で行っています。

textOutput関数は簡単で、cards(0)～cards(n)までのカードをreadCard関数（これはデータベース本体で使用しているもの）で順に読み込んでいきます。データはData配列にセットされますから、Data(0)～Data(30)をCR+LF付きで出力していくだけです。ひとつのカードを出力したら、改頁+CR+LFを出力してカードの区切りとし、次のカードにとりかかります。プログラムを見ればなにをやっているのかすぐにわかると思います。

テキストデータの入力

テキストデータを読み込むほうも、データベース本体で使用している関数を使えば簡単にケリがつきます。基本的な方針としては、Data配列にテキストデータを読み込み、それをwriteCard関数で出力すればOKということになります。

textInput関数を見てください。textOutput関数同様に、こちらもnfpをパラメータとして受け取ります。nfpは今度はテキストを読み込むファイルのファイル番号を出しています。読み込める最大のカード数は全カード数（100枚）から現在使用中のカード数（usingCards枚）を引けば求められます。先月はcommands関数中にusingCards変数をとっていましたが、textInput関数でもusingCardsを参照できるようにこれを大域変数にすることにします。具体的にはcommands関数からusingCardsの宣言を削除し、プログラム先頭の大域変数宣言にusingCardsを付け加えてください。

textInput関数は続いてdbasep配列に－1をセットします。読み込むテキストファイルから新しいカードを作るのですから、dbasep配列を未使用の状態にするわけですね。同時にData配列にも""をセットして未使用にします。

あとはnfpから1行ずつデータを読み込み、dataEntry配列を参照しながらデータをセットすべき行に読み込んだデータをセットする作業を繰り返します。改頁を読み込んだらカードはそこで終了。newCard関数で新しいカード番号をもらい、writeCard関数でカードを書き込みます。usingCardsは当然ひとつ増えますね。

Toolsメニューの追加

今回作成したテキスト入出力は、データベース操作中にいつでも使用できるようにしておくのが便利です。そこで、データベースメニューにToolsというメニューを追加しておきましょう。commands関数でメニューを表示しているところにToolsを書き加え、続くswitch文の中に、

```
case 't'
case 'T'
  tools( chosen )
  break
```



という選択肢を加えておきます。

さてtools関数ですが、これは

OutputText　InputText >

というメッセージを表示する、commands関数と同じようなメニュープログラムになっています。パラメータとして渡されたchosenは、cards配列に何枚のカードが登録されているかを表し、これはtextOutput関数で利用されます。例によってメニュー選択のキー入力はswitch文によって場合分けされ、それぞれの処理が行われます。追いかけてみてください。

ディスクに収めたcbase.xには、このテキスト入出力に加え、カード設計の変更をツールとして用意してあります。これでよしと思ったはずの設計も、データを入力していく途中で不都合を感じて設計し直したくなる場合があります。すでに入力したデータは無駄にしたくない、この要求に応えるものです。今回のリストに入れなかったのは、予想以上に大きくなってしまったからです。使い方はリファレンスマニュアルを見ていただくとして、同様の機能を持ったものを自分で作ってみるというのはいい勉強になるだろうと思います。

最後に

X-BASICの人気が徐々に盛り上がりつつあるようです。最初は既存のBASICとはまったく異なったBASICであり、しかも動作速度が遅いということでないがしろにされがちでしたが、C言語を扱うより容易にプログラムが作れること、コンパイルすれば十分実用になる速度を得られることが認められてきたためでしょう。かく言う私も連載を始めるまではほとんど使ったことはありませんでした。それが連載を続けているうちに、まずはX-BASICでプログラムを作ってみて、それからCで作り直すなり、コンパイルしてからC言語の段階で手を入れるというアプローチをとるようになってきました。

プログラムの動作を確かめるのにこれほど手軽な言語はありません。しかも、Yetのようなリアルタイム性を備

……ゲームから、カード型データベースという実用的なアプリケーションを作るほどの能力も備えています。これからもプログラミングの友としてX BASICを使っていきましょう。X BASICの魅力をお伝えする機会が与えられたこと、そして連載につきあってくださった読者の皆さんに、心よりの感謝を捧げたいと思います。

リスト1　データベース用追加ツール

```
10000 /*
10010 /*  追加ツール
10020 /*
[illegible]
    while flag
      locate 0,0
      print "OutputText  InputText ? " [illegible]
      c$ = inkey$
      switch asc( c$ )
        case 27                    /* ESC
          flag = 0
          break
        case [illegible]
        case [illegible]
          file$ = getLine( "出力ファイル名：" )
          nfp = fopen( file$, [illegible] )
          textOutput( nfp [illegible] )
          fclose( nfp )
          break
        case [illegible]
        case [illegible]
          file$ = getLine( "入力ファイル名：" )
          [illegible]
          nfp = fopen( file$, [illegible] )
          [illegible]
          if nfp = -1 then {
            locate 0,0
            print file$ + "が存在しません。",chr$(1)
            beep
            [illegible]
          } else {
            textInput( nfp )
            fclose( nfp )
          }
          break
      endswitch
    endwhile
endfunc
/*
/*  ファイルへのテキスト出力
/*
func textOutput( nfp, n )
  [illegible]
  for i=0 to n-1
    readCard( cards( i ) )
    for j=0 to 79
      if dataEntry( j ) [illegible] then {
        fwrite( Data( j )+chr$(13)+chr$(10), nfp )
      }
    next
    fwrite( chr$(13)+chr$(10), nfp )
  next
endfunc
/*
/*  ファイルからのテキスト入力
/*
func textInput( nfp )
  int i, j
  for i=usingCards to 99 [illegible]
    if feof( nfp ) then break
    [illegible]
    for j=0 to 99
      [illegible]
      Data( j ) = ""
    next
    for j=0 to 79
      if j < 0 then {
        if dataEntry( j ) = 255 then {
          continue
        }
      }
      [illegible]
      if [illegible] then break
      locate [illegible]
      print [illegible]
      Data( j ) = [illegible]
    next
    writeCard( newCard() )
    usingCards = usingCards + 1
  next
  readCard( cards( 0 ) )
endfunc
/*
/*  1行入力
/*
func str getLine( message$ )
  [illegible]
  locate 0,0
  print [illegible]
  [illegible]
  while [illegible]
    [illegible]
  endwhile
  return( tmp )
endfunc
```

カード型データベース CBASE.X リファレンスマニュアル

CBASE.Xの動作環境

Human68kのコマンドモードまたは、VS2.X（付録ディスク1収録）によって拡張されたビジュアルシェル上で動作します。

ただし、付録ディスク1には容量の都合で日本語フロントエンドプロセッサASK68K.SYSは収録されていません。解凍したディスク1を利用して日本語入力を行う場合には、以下の手順に従ってASK68K.SYSを読み込んでください。

1) お手持ちのシステムディスクからASK68K.SYSをコピーする。

ASK68K.SYSは、X68000付属のシステムディスクのSYSというディレクトリに入っています。ドライブ0からシステムディスクを起動し、コマンドモードに入ってください。ドライブ1にディスク1を入れて、

A>COPY A:¥SYS¥ASK68K.SYS B:¥SYS

と入力しリターンキーを押します。

2) CONFIG.SYSにASK68K.SYSを登録する。

まず、エディタED.Xを起動します。

A>ED B:¥CONFIG.SYS

CONFIG.SYSの内容が表示されたら適当な位置に次の1行を追加します。

DEVICE = ASK68K.SYS B:¥X68K_M.DIC B:¥X68K_S.DIC

これは、辞書をBドライブで使用する場合です。

ESCキーに続いてEキーを押すとセーブしてエディタを終了します。

以上で準備は完了しました。ディスク1をドライブ0、辞書ディスクをドライブ1に入れてリセットボタンを押してください。

＊

【起動方法】

cbase 〔データベースファイル名〕〔0〕

・データベースファイル名は拡張子を付けずに指定する。データベースファイル名を省略した場合は、起動後ファイル名の入力となる。

・オプションとして0を指定すると、VS2.Xから起動した場合と同様に12ドットフォントで表示される。ただし終了後カーソルは表示されない。

【CBASE.Xの能力】

CBASE.Xは最大で2000データ、250枚のカードを扱うことができる。この講座で作成したデータベースよりカード数を150枚増やしてあり、設計情報ファイル（*.fml）を除いてファイルに互換性はない。ただし、CBASE.XではX BASIC版データベースで作成したファイルも扱うことができる。この場合カードの最大数は100枚である。

【データベースの設計】

起動時あるいは起動後に指定したデータベースファイルが存在しない場合、新規データベースと見なしてデータベースの設計画面になる。設計画面で使える編集キーは以下のとおり。

←	カーソル左
→	カーソル右
↑	カーソル上
↓	カーソル下
BS	カーソル左の文字を消去
DEL	カーソル位置の文字を消去
ROLL UP	カーソル行を消去し、以下の行を1行上へ
ROLL DOWN	カーソル位置に1行挿入し以下の行を1行下へ

データ入力位置は#で指定する。データ入力位置は1行1カ所に限られる。たとえば

名称　　#

という行を作成すると、表示時には、

名称　　■

のようにカーソルが#位置に表示されデータ入力できるようになる。

データベースの設計は、ESCキーを押すと終了する。データベース用ファイルの作成が開始され、しばらく待つとデータ閲覧画面になる。

【データの閲覧】

データ閲覧画面では、画面最下行に

0 / 30 (40) Edit……

という形式のメニューが表示される。カッコ内の数字は現在使用中のカード数、その左の数字は使用中のカードから閲覧用にピックアップされているカード数、最も左の数字はピックアップされたカード中の何番目のカードを表示しているかを示している。カードの閲覧はROLL UP、ROLL DOWN……ピックアップされているカード以外は閲覧できない。

メニューは最初の英大文字をキー……入力することによって選択される。以下にその機能を挙げる。

●カードの修正（E　Edit）

表示されているカードへのデータ入力、データ修正を行う。ESCキーを押すと入力・修正は終了する。Editメニューではカーソル上下左右、リターン、BS、DELキーが編集用に使用できる。入力された文字は常にカーソル位置へ挿入される。

●カードの削除（D　Del）

表示されているカードを削除する。削除の確認を求め、キーが押された場合のみ削除を行う。

●ピックアップカードの設定（C　Clear）

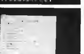
ピックアップカードを無効とし、再び使用中の全カードをピックアップカードとする。

●カードの除外（X　eXclude）

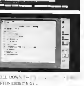
表示中のカードをピックアップカードから外す。カードの削除ではない。



標準的な動作モード。ディスク1のサンプルデータindex99.datの場合には、コマンドモードなら cbase index99 で起動する



……フォントの文字を使用するモード。cbase index99 0 で起動する



ビジュアルシェル上で動くモード。解凍したディスク1では index99.……のアイコンをダブルクリックすると起動する

■カードの検索（F　Find）

条件を指定してカードの検索を行う。選択すると、

　Normal, Additional >

というメニューが表示される。これは検索モードの指定で、Normal検索はピックアップされたカードの中から条件に合致するものを探し出しピックアップカードとする。Additional検索は、ピックアップされていないカードの中から条件に合致するものを探し出しピックアップカードに加える。

検索モードを指定すると条件入力画面となる。条件入力画面はカードの修正画面と同様の画面で、データ入力位置でカーソルが点滅している。検索したいデータが入力されている位置までカーソルを移動し、条件を、

　「演算子」「比較データ」

の形で入力する。演算子は以下のとおり。

=	データ	=	比較データ
<	データ	<	比較データ
>	データ	>	比較データ
!	データ	≠	比較データ
(	データ	≦	比較データ
)	データ	≧	比較データ

（データは検索条件が入力された位置に入っているデータ）

例）　重量　　<600

　重量が600より小さいカードを探し出す

また、特定の文字列を複数の場所から探し出す場合に、何度も同じ条件を指定する面倒を省略するため、

　^　　上と同じ条件

という演算子が用意されている。この場合比較データは必要ない。比較はすべて文字列比較で行われるので注意されたい。

検索条件の入力はESCキーで終了する。条件入力が終了すると、検索方法が画面最下行に表示される。

　And, Or >

Andは入力したすべての条件を満たすカードを検索する。Orは入力したすべてのいずれかを満たしているカードを検索する。

■カードの並べ替え（S　Sort）

ピックアップされているカードの並べ替えを行う。昇順・降順の並べ替えが可能で、検索条件を入力したときと同様に、並べ替えのキーとなるデータが入力されている場所に「<　昇順」「>　降順」を書き込みESCキーを押すと並べ替えられる。複数キーによる並べ替えはできない。最初に指定された場所だけが有効である。

●ツール（T　Tools）

テキスト入出力、カードの再設計を行うツール選択メニューを起動する。

●終了（Q　Quit）

カード型データベースを終了する。

【ツールの使い方】

Toolsメニューによってツールが起動されると、次のメニューが表示される。

　OutputText, InputText, EditFormat >

ESCキーで終了。以下にその使い方を説明する。

■テキスト出力（O　OutputText）

データベースファイルを、ワープロやエディタで利用可能なテキストファイルとして出力する。ファイル名を入力すると、各カードのそれぞれのデータを改行で区切り、カードを改頁で区切ったテキストファイルが生成される。出力されるのはピックアップされているカードだけである。

■テキスト入力（I　InputText）

データを改行で、カードを改頁で区切ったテキストファイルを読み込み、データベースファイルに追加する。ファイル名を入力すると、データの読み込みが行われる。読み込まれたデータはピックアップカードには自動追加されないので、Clearメニューなどを利用してピックアップする必要がある。

●カードの再設計（E　EditFormat）

EditFormatを実行するとカードの設計画面が再表示され、再設計用メニューが画面最下行に現れる。

　Move, Edit, Update >

再設計はESCキーで終了する。

Moveは行の移動を行う。最初に、

　移動するレコード指定

と表示されるので、反転表示された行をカーソル上下で移動し、動かしたい行でリターンキーを押す。次に、

　移動先指定

と表示されるので、同様にして移動先を指示する。指定された行は削除され、移動先に挿入される。ESCキーで再設計メニューに復帰する。

Editは行の修正を行う。カーソル左右、BS、DEL キーが編集用に使用できる。修正はリターンあるいはESCキーで終了し、再設計メニューに復帰する。データ入力位置を示す#を削除した場合は、Updateするまでその行に#を書き込むことはできない。

Updateは設計の更新を行う。行の移動、#の削除に対応して入力されているデータの移動・削除も同時に行われる。誤って設計情報から#を削除してしまった場合はUpdateせずにESCキーでツールメニューに復帰すれば元の設計情報が復帰する。

【付属データ】

サンプルデータファイルとして、年末インデックスの一部がディスクに付属している。収録されているのは特集と連載だけであるが、これを参考に自分のデータベースを構築してみていただきたい。カードの設計方法は、

　Tools EditFormat

で表示されるインデックスの設計を参照されたい。

シミュレーションへの道も絵描きから

Kamon Masato　華門 真人

前回X1 BASICで記述していた高速道路の料金所シミュレーションプログラムをX68000のX-BASICで記述してみましょう。また耳に馴染んでいない言葉が出てくると思いますので、事前に前回の記事を読み返しておくことをおすすめします。

Experience

昨年の春（というとまぎらわしいが1990年の春のことである）、僕はアメリカに住んでいる両親のもとを訪ねた。そこでひとつの面白い事実を発見した。

発見は例によってクルマに乗っているときのことである。クルマとはいってもアメリカでのことであるから、もちろんアルシオーネではない。そのときはビュイックのパークアベニューという、いわゆるフルサイズのアメ車、しかも10年くらい前の古き良き時代のに乗っていた。親父の会社のクルマを借りていたのだ。

言い忘れていたが、僕の両親はテキサス州のヒューストンに住んでいる。ヒューストンというと夏に開催されたサミットのことを思いおこされる博識な方もいらっしゃるだろう。確かにヒューストンはサミットシティとして一時的には有名になったが、実際のところはあまり知られていないように思う。

実はヒューストンはブッシュ大統領の地元であり、全米第4の大都市でもある。ほら、驚いたでしょう。かくいう僕自身も実際に行ってみるまでは、テキサスというぐらいだから砂漠の中にぽつんとある街なのかと思っていた。

実際のところは緑が豊かで広々とした都会であった。いやあいいところですよ、本当に。ただこれはアメリカのほとんどの都市にいえることだけれども、あまりにも広々としているうえに公共交通機関もほとんどないから（バスぐらいだ）、どこに行くにも、なにをするにもクルマがないと生活できない。

というわけでビュイックにはお世話になった。これではひとりに1台クルマがないと生活していけないなと実感した次第。しかし逆にクルマがそれこそ「ゲタ」代わりになっているだけあって、道路網は素晴らしく発達している。一般道はたいてい片側3〜4車線と広々しているし、高速道路も網の目のように張りめぐらされている。日本でいうと首都高速のようなものだけれど、どこでも最低片道3車線はあるし、しかも大部分は「タダ」。

まったく羨ましくなってしまうぐらいの充実ぶりである。特に僕のようなクルマ好きには。でもそんな広々とした道路も、ラッシュアワーには大渋滞している。さすがはクルマ大国というべきか。

本題に戻ろう。僕が興味深い発見をしたのは、ガルベストンというヒューストンから50マイル（約80km）ぐらい離れた港町に遊びに行った帰りのことである。行きは下道でのんびりと行ったのだが、帰りは疲れていたこともあってI-45（インターステート45号線）で素直に帰ろうということになった。

I-45は快適そのものであった。なんといってもアメリカ、道路は広いし、こちらはまるで戦艦のようなアメ車である。乗り心地は抜群……と、前方で工事のようである。ああ、やっぱり日本と同じでアメリカでも道路工事ぐらいあるのだ。しかし不思議なことに、クルマの流れが悪くなるようなことはない。

すぐにその理由がわかった。工事区間に近づくにつれ徐々に車線が狭くなっていくのである。もともとアメリカの車線はゆったりしているのだが、それがあっという間にクルマぎりぎり1台分にまでなってしまった。

しかしその反面、車線は減少しない。だから合流などによって渋滞することもないということだったのだ。

ここでようやく問題は核心にせまる。日本ではよく工事で渋滞してしまうことが多い。いつもはすいている道路がその日に限って大渋滞、なぜなんだろうと思っていたら実は工事だった、という経験をした人も多いと思う。

Observation

同じ工事でもアメリカでは渋滞はしないのに、日本では渋滞する。違いは道路のキャパシティ（容量、噛みくだいていうならば処理能力）にある。アメリカでは車線1つひとつの幅は狭くなっても車線の数自体は減らない。だから道路のキャパシティは工事中でも減ることはない。

ひるがえって日本では、すぐ車線規制をして、車線数を減らしてしまう。こうなると工事によってキャパシティが減ってしまうことになり、結果どんどんやってくるクルマをさばききれなくなって渋滞が起こる、というわけだ。

もっと身近な例でいえば、スーパーマーケットのレジなんかを考えてみてもいい。レジがあまりあいていないと、数少ないレジにお客が集中するものだから、レジに長蛇の列なんてことも珍しくない。

もっともそんなことしゃあお客様商売としては失格だよね。そこで店のほうもレジの数を増やすなどして対処しようとするわけ。すると列がたちまち短くなったりする。

結局これはレジのキャパシティの問題といえる。レジの数が少ないと、当然同時に処理できる客の数が少ないということだから、言い換えればキャパシティが小さいということになる。

ところがレジの数を増やせば、同時に多くの客を処理することができる。それがすなわちキャパシティが大きくなる、ということなわけだ。

もっとも、混むかどうかというのはキャパシティだけによるわけではない。処理対象がどれだけあるか、ということにも大きく左右される。

これは少し考えてみればわかることだろう。たとえば1時間に2、3人しかお客のこな

い・スーパーであれば、どう考えてもレジは1台あれば十分だろう（スーパーというよりも個人商店というべきか）。

ところが1時間に何百人、何千人単位でお客がくる大きなスーパーでは（ちなみに最近はこういった規模の大きいスーパーをハイパーマーケットと呼ぶらしい。うーんハイパー）、相当多くのレジを設置しないと、処理が追いつきそうもないというのは自明だろう。

さらに付け加えるならば、キャパシティは簡単な計算によっても考えてみることができる。前回の料金所モデルで考えてみよう。まず改良前のモデルの1分間を想定する。平均すると、この1分間に20台のクルマが流入してくる。このうち18台が右ハンドルで、2台が左ハンドルである。2台の左ハンドルは発券機3で14秒あれば処理できる。

ところが右ハンドルはというと、発券機2台をもってしても63秒（=18×7－2）かかってしまう。すなわち1分間に流入してくるクルマを、1分間では処理しきれないのである。つまりパンク状態。大渋滞を起こすのも無理はない。一方改良後は、同じ18台の右ハンドルでも、発券機3も利用することにより処理しきれてしまう。

とまあ、僕の観察したちょっとした事実から渋滞発生の理論にまで話を拡大してみた。おそらく日本では安全第一ということで車線規制をするのであろうが、こんなことからも日本人とアメリカ人の考え方の違いが垣間見えて面白いものである。こう考えてみるとなかなか渋滞っていうのも身近なものに思えるから（あまりありがたくない身近なものではある）。

ところで、アメリカでの話にはちゃんとしたオチがある。うん、これならば渋滞にはならないな、と感じしたのはいいのだが [illegible] 忘れてはいけない、僕がその [illegible] たのはアルシオーネではな [illegible] でいうなら超ヘビー級のビュイックだ。

皆さんはいわゆるフルサイズのアメ車に乗ったことがあるだろうか。これがとにかく大きいのである。残念ながら正確な大きさは忘れてしまったのだが、そのボリューム感はベンツ560（いわゆる大ベン）をも凌ぐものがある。

もっとも僕でさえはじめは尻込みしてしまった巨大なビュイックであるが、いったん乗りだしてしまえば特に困ることもなかった。なぜならアメリカでは道路が十分に広いからである。

しかしそれも道路が広いときの話。ひとたび工事となると、あれほど広かった車線も急激に狭くなる。巨大なビュイックはほとんど車線をはみださんとする。ドライバーとしてはとにかく車線をはみださないようにするだけで精一杯。そこを左右から他のクルマが追い越しをかけていく。

はっきりいってあのときは [illegible]

リスト1

```
1000 /* Simulation model 2 ver 6 01
1010 /*
1020 /*         for X-BASIC
1030 /*
1040 /*              1000 10 [n] Common
[illegible]
```

にしてみれば大変な騒ぎだったのである。が、あとで同乗者に聞いてみたところ、なにも気がついていなかったようである。羨ましい。

このような「恐怖の」経験をしたのは僕だけではない。どこかのクルマ雑誌で、ドイツ現地取材という記事があったのだが、ドイツでもやはり工事の際は車線幅を狭くして車線数を維持するらしい。その記事の筆者によると「とても怖かった」ということだ。

Groundwork

さて、前置きはこれぐらいにして本題に入る。前回はコンピュータシミュレーションの有効性について見てきたわけだが、約束どおり、今回はどのようにしてプログラムにするか、ということを考えていこうと思う。

題材としてはリスト1を用いることにする。これは前回のモデルをX-BASIC上で実現させるためのもの。先月のX1用と基本的に一緒と考えてもらって差し支えない。

念のためにモデルの解説をしておこう。モデルの舞台となるのはある架空の料金所である。この料金所での通行券の発券をシミュレートする。

シミュレーションの前提は以下のとおりである。

・発券用の車線は3車線あり、そのうち2つは右ハンドル専用（第1 2レーン）、残りのひとつが左ハンドル専用（第3レーン）である。

・発券にはどの発券機でも7±2秒かかり、クルマは3±2秒間隔でやってくるとする。なお、クルマ全体に占める左ハンドルの割合は10%であるとする。

・料金所に入ってくるクルマは左ハンドルならば無条件に第3レーンに進む。右ハンドルは、第1レーンか第2レーンのうち、すいているほうに進む。もし同じように混んでいるときは等確率で第1レーンか第2レーンのどちらかに進む。

あるシミュレーションモデルをプログラムとして実現する際、まずなにから始めればいいのだろう。当たり前のことかもしれないが、まずモデルをしっかり把握することから始めなければならない。

まずはモデルを図にして要点を把握することだ。まさしくシミュレーションの道もお絵描きから始まるのだ。図1（前回の図3と同じもの）がこれにあたる。トランザクション（処理対象）の動き、あるいはファシリティ（処理する施設）との関係などをしっかり把握しよう。

把握できたところでフローチャートを作成する。そしてそのフローチャートをもとにプログラムにする。そう、普通のプログラムを作成するのと基本的には変わらないのだ。

もっとも、フローチャートといっても少し独特なものを利用することにする。それでは図2を見てほしい。これはトランザクション（このモデルの場合、クルマ）を中心にしてモデルの流れを言葉で書いてみたものである。

まず初めに料金所にクルマが流れ込んでくる。ただ、クルマといっても右ハンドルと左ハンドルの2種類ある。そこで発券機の選択を行う必要がでてくる。

すなわち、左ハンドルならば無条件に発券機3を選択する。一方、右ハンドルならば、発券機1か2のどちらかを選択することになる。そこで右ハンドルの場合のみ発券機1、2の選択という項目が加わることになる。

さて、クルマごとに発券機が決まった。ここでようやく本来の目的である「発券」という作業ができることになる。ただ、ひと口に発券とはいっても、場合によっては待ち行列のことなども頭に入れておかなければならない。

かくのごとき手順を経てようやく、クルマを料金所から解放することができる。もちろんその前に、結果を得るための計算などもそつなくこなしておかねばならない。

図1　料金所の様子

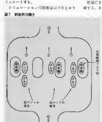
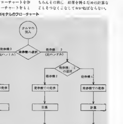


図2　料金所モデルのフローチャート



以上が図2のフローチャートの概要というわけだ。こうして言葉で表現してみるとモデルを的確に、しかも容易に把握することができる。何事も最初はシンプルにという法則は、ここでも生きているのだ。このシンプル極まりないチャートが、実際にプログラムを作るときの土台になる。

以下では、図2とリスト1を対比させながら説明していくことにしよう。多少個所に違いはあるが、先月のX1版でも基本は同じである。

main {

およそプログラムであるかぎり、まずは初期設定というものが必要になる。まあよくいわれるところの「おまじない」みたいなもの。リスト1では2000行台がこれに該当する(以下、行番号は特に指定がない限りリスト1を指す)。

やっていることは変数の宣言と初期値のセットぐらいのもの。参考までに変数名のだいたいの意味を述べておこう。まず変数名にpが含まれている場合はパラメータを表している。同様にしてxactはトランザクションに関する変数を、qは待ち行列(キュー)に関する変数を指している。

残念ながら筆者の「行きあたりばったりな」性格(?)を反映してか、あまりわかりやすい変数名にはなっていないが、これでも参考にはなるだろう。

付け加えておくと、X-BASIC版では実行前に目的のトランザクション数を入力できるようにした。こうしたほうがCでコンパイルしたときに使いやすいからなのだが、お望みであればX1版でも同様に改造できることはいうまでもない(2090行。もちろんX1版のを書き換えるだけである。念のため)。

さて、おまじないもすませたところで、いよいよメインルーチンに入ろう。図2の頭と終わりをつなげると、メイン部はループになる。そこで3000行台のようなループを作ってみる。このループを終了する条件はただひとつ、処理したトランザクションの総数が目的のトランザクション数に達した場合である。そこで、3020行と3120行で無限ループを作り、3100行でループから抜けるようにした、という次第。

さて、次はメインルーチン内でどのような処理を行うか、ということを考えてみよう。図2を見ると、メインルーチン内は大きく2つのパーツに分かれている。すなわち、「トランザクションを発生させ、料金所に割りあてる」パーツと、「待ち行列などの、ファシリティでのトランザクションの処理と計算を行う」パーツである。

これら2つのパーツをそれぞれサブルーチンとして処理してしまうと、メインルーチンは非常に単純になる。もうおわかりだと思うが、この2つのパーツは前者がgenerate()、後者はqueue()となっている。

以上を頭に入れて、メインルーチンの筋道を追ってみよう。まず、時刻を調べ、新しいトランザクションを発生させるべきかをチェックする。もし新しいトランザクションを発生させるのであれば、generate()を呼ぶ。

次に、各ファシリティ(1~3)に対しqueue()をコールする。このqueue()で待ち行列およびファシリティのコントロールを行うわけだ。この時点で処理されたトランザクションの数が目標値に達すると、ループから抜け出す。

しかしたいていの場合は時間を経過させたあと、ループの頭に戻ることになる。

ここで疑問を感じた人。あなたは鋭い。というのもこのモデルは離散変化モデルのはずだからである。このループでは時間を中心にモデルが動いていっているように見えないだろうか。もしそうならば、これは連続変化モデルになってしまう。

確かにこのモデルは離散変化モデルだし図2もそのように書かれている。前回で詳しく述べたように、離散変化モデルでは時間はモデルが変化する重要なポイントをつなげたものとして表現される。このモデルの場合、このポイントとは、

ポイント1
―Xactが料金所に流入
(待ち行列に並ぶ)
ポイント2
―ファシリティの使用を開始
(待ち行列を抜ける)
ポイント3
―ファシリティの使用を終了
(料金所を抜ける)

の3つにしぼられる。

すなわち、このモデルでは上記の3点でのみ時間が刻まれるはずだ。ところがプログラム上では、あたかも連続変化モデルのごとく時間を連続的に変化させているように見える。

はたしてこのギャップはどこから生じているのだろうか。ポイントはファシリティが複数存在しているという点にある。もしファシリティがひとつしかないのであれば、時間の変化ポイントがくるたびに、時間tに間分を加えるかたちで時間を変化させればよい。これが「本来の」離散変化モデルであるし、問題はまったくない。

ところがファシリティが今回のように複数になると、話は異なってくる。すなわち各ファシリティごとに現在時刻が異なってきてしまう、という問題が生じる。つまりこういうことだ。ファシリティ1、2、3それぞれでそれぞれのポイントに従って独自に時間を変化させていく。そうすると当然、ある時点で各ファシリティの様子を見てみると、各ファシリティでそのときの時刻が違ってくる。

これは実に不都合な状態である。たとえばある時刻に新しいトランザクションが発生したとする。そのとき、どのファシリティの待ち行列に並ぶかを決めようにも、各ファシリティで時間が違うから、その時刻の各ファシリティの様子を知ることができない。その結果トランザクションがどのファシリティに属するかも決定できない、という事態になってしまう。

そこで今回のようなプログラムでは、便宜的に時間を中心にループをまわすという方法をとるというわけ。しかし、これでも離散変化モデルである以上、連続変化モデルとは決定的に異なる点がある。

すなわち、ループ内でこそ時間tは1ずつ増加していくが、ファシリティの観点に立ってみると、時間が変化するのは変化ポイントのときだけなのである。これはあとで各ルーチンを見てみるとよくわかると思う。

とりあえずそれは置いておくとして、ループを抜けたあとのことにも触れておこう。といっても単にresult()で結果を表示しているだけのことなのだが。

generate

それでは次に、各ルーチンを見ていくことにしよう。まずはトランザクションを発生させ、各ファシリティに割りつけるgenerate()から。リストでいうと10000行台になる。

このルーチンはトランザクションを発生させるべき時間になったら呼ばれる。そのため、ルーチン内ではまず、その次に発生させる時間をセットしなければならない。ここで、次に発生させる時間は現在時刻+(パラメータをもとに乱数で決まる発生間隔)で表される。10030行がこれである。

次回への布石をすませたら、やるべきは発生したトランザクションのファシリティへの割りつけである。そのためには乱数によって右/左ハンドルを決定する。もし左ハンドルであれば、ファシリティは3に決定(pl=3)である。

ところが、右ハンドルの場合は、ファシリティ1/2の混み具合を見たうえでどちらかを選択しなければならない。これを一手に引き受けているのがgensub()というわけだ。gensub自体は単に条件を羅列しているだけだから、自分でフォローしてみてほしい。

queue

お次にひかえますは、待ち行列やファシリティの状態をコントロールする関数queue()である。queueはファシリティごとに実行されるわけだが、まずそのファシリティで新たにトランザクションが発生したかどうかをチェックする。

トランザクションが発生していたらmkque()に飛ぶ。mkqueでは現在ファシリティが使用中かどうかを調べなければならない。使用中であれば話は簡単。待ち行列の最後尾に加えてやればいいのだ。

使用中でなかった場合、この場合、処理はmkqfr()にまかされるが、やっていることはファシリティでの処理時間が何秒になるかを決定するだけの話である。この処理時間もパラメータをもとに乱数で決定される。もちろんファシリティを使用中に変更してトランザクションの発生処理は終わる。

別に、トランザクションが発生しておらず、また、ファシリティが使用中でもない場合はなにもせずに帰ればよい。ファシリティが使用中でも、まだ使用が終わらない場合も同様である。話が少し複雑になるのは、ファシリティの使用がちょうど終わった場合である。

この場合もそのファシリティに待ち行列が存在しているかどうかで、話が変わってくる。まずは待ち行列が存在しない、すなわち後ろに誰も待っていない場合を考えてみよう。こちらは話が簡単で、ファシリティを空きにして(u(i)=0)リターンする。

一方、待ち行列が存在している場合は、次の者へ順番を回してやらなければならない。そして新たにファシリティを使用することになったトランザクションに対し、mkqfrで使用時間を設定する。さらに最後に待ち行列の長さを1だけ減らしてリターンする、というわけである。

付け加えるとこのqueueルーチンでは、このような処理と同時に、結果を得るための計算も行っている(というか、こちらが本題ともいえるのだけれど)。これに関しては表1を参考に自分で考えてみるといいだろう。

さて、こうやって見てきて、サブルーチン内ではモデルの重要変化ポイントで時間が刻まれているということ、おわかりいただけましたか?

Result

こうして丁寧に見ていくと、意外とシミュレーションプログラミングというものが簡単だということがわかるだろう。意外かな。一見複雑そうに見えるシステムもうまくモデル化してやれば簡単にシミュレートすることができるのだ。

ところで、前回はこのモデルにさらに改良を加えてみた。すなわち、第3レーンを右ハンドルでも左ハンドルでも利用できるように変更した。X-BASIC上でも同じように改良してみることにしよう。リスト2をリスト1につぎはぎすれば出来上がりだ。行番号でいうと、10100行から10190行を書き換え、30000行以降を加えればいい。これはルーチンでいえばgensubを書き換えていることになる。すなわち右ハンドル時のファシリティ選択ルーチンだけを書き換えているというわけ。

それでは改良前と改良後、それぞれを実行してみることにしよう。今回はトランザクションの目標値を10000として実行してみた。これもX-BASIC版ではコンパイラを通すことによってCでコンパイルすることができるからである。

前回のX1版を実行してみて実行速度に不満を持たれた方も多いであろう。確かにあれは遅い。とてもじゃないが目標値1000以下でしか実行する気になれない。あれもいろいろとキタナイことをすれば多少は速くなるのだろうが、多少の速さよりプログラムの素直さを重視させてもらった。

そのストレスを解消すべく、今回のプログラムをコンパイルして実行したものは速い。これならば気長に10000回待ってもいいな

表1 統計計算式表

▶average utilization

$$=\frac{\text{(ファシリティ使用時間総計)}}{\text{(全シミュレーション時間)}}$$

= suttl(i)/t

▶average time/Xact (facility)

$$=\frac{\text{(ファシリティ使用時間総数)}}{\text{(ファシリティで処理したXact数)}}$$

= suttl(i)/dcnt(i)

▶average contents

$$=\frac{\Sigma\{\text{(待ち行列に入っていた時間)}\times\text{(トランザクション個数)}\}}{\text{(全シミュレーション時間)}}$$

= quett(i)/t

▶percent zero

$$=\frac{\text{(zero entries)}}{\text{(total entries)}}$$

= nque0(i)/nque(i)

▶average time/Xact (queue)

$$=\frac{\Sigma\{\text{(待ち行列に入っていた時間)}\times\text{(トランザクション個数)}\}}{\text{(total entries)}}$$

= quett(i)/nque(i)

リスト2

```
10100 func char gensub()
10110   char p
[illegible]
10190 endfunc

30000 /*
30010 /* [illegible]
30020 func char [illegible]
[illegible]
30180 endfunc
31000 /*
31010 /* generate [illegible]
31020 func char [illegible]
31030   char a
31040   switch a
31050     case 0 : [illegible] : break
31060     case 1 : [illegible] : break
31070     case 2 : [illegible] : break
31080     case 3 : [illegible] : break
31090   endswitch
31100   return(p)
31110 endfunc
```

なという気にさせてくれる。さすがはCコンパイラ　ありがたや。
　まあそれはともかくとして、実行結果を見てみよう。表2が改良前、表3が改良後である。目標部を大きくしても傾向は変わらないことがわかるだろう。

and……

　以上でどのようにしてプログラムを作成するか、ということを見てきた。都合と都しいな、と感じた方もいらっしゃるだろうし、なんだ簡単じゃない、と思われた方も多いだろう。もちろんそれは個人差なのだけれど、仮に簡単だと思った人でも、モデルを大幅に改変しようとするとそれなりの苦労をすることになるだろう。
　え、でも今回の改良は簡単だったじゃない、だって、それはこのプログラムができるだけフレキシブルに書いてあったからの話。とはいえフレキシブルにも限界がある。ある程度以上の変更をしようとすると、なかなか大変な事態になるだろう。
　たとえばの話、この料金所に救急車のような緊急車両がやってきたらどうするか。当然待っているクルマを追い越して緊急車両を通過させなければならない。すなわち待ち行列に対する「割り込み」が生ずることになる。ここまでくると今回のプログラムで対処することは、不可能とはいわないまでもかなり難しいだろう。
　それではどうするか。ひとつの一般的な道として「それじゃあシミュレーションを実行するための専用言語を作っちゃえ」という考え方がある。このような考え方をした人は多いらしく、有名な言語としてGPSS、SIMSCRIPT、SOL、そしてSIMULAなどがある。
　これらの言語は実は歴史が長く、1961年にはもうGPSSそしてSIMSCRIPTが開発されている。このなかでもいちばんメジャーなのがGPSS (General Purpose Simulation System)である。
　ちなみに今回のモデル（改良前）をGPSSで書いてみるとリスト3のようになる。ひと目見ればわかるとおり、たったの20行ほどである。BASICだと簡版でも100行はかかるのに、である。もっとも、それが専用言語の専用言語たるところなのだろうか。
　専用言語のよさはプログラムが簡単に書けるという点にだけあるわけではない。その汎用性の高さもその特徴のひとつといえる。たとえばの話、前回の「割り込み」なども軽くこなす。モデル内部での細々な作業にも気を配らなければならないBASIC版に対し、モデルの構築だけに集中できるのが専用言語のいいところといえるだろう。
　さて、いいことずくめの専用言語なのだが、残念ながらX68000上には存在していない。これでいいのだろうか。

NEXT

　もちろんこれでいいわけがない。この連載ではX68000用のシミュレーション専用言語を紹介する予定である。おおそうか来月か、と焦ってはいけない。

表2　リスト1の実行結果

```
Result

Simulation time (t) : [illegible]

-- Facility --
name     average       number     average
or no    utilization   entries    time / Xact
1        [illegible]   [illegible] [illegible]
2        [illegible]   4400        [illegible]
3        [illegible]   [illegible] [illegible]

-- Queue --
no   max        average    total     zero      percent
     contents   contents   entries   entries   zero
1    [illegible] [illegible] [illegible] [illegible] [illegible]
2    [illegible] [illegible] [illegible] [illegible] [illegible]
3    [illegible] [illegible] [illegible] [illegible] [illegible]
```

表3　リスト2の実行結果

```
Result

Simulation time (t)   [illegible]

-- Facility --
name     average       number     average
or no    utilization   entries    time / Xact
1        [illegible]   [illegible] [illegible]
2        [illegible]   [illegible] [illegible]
3        [illegible]   [illegible] [illegible]

-- Queue --
no   max        average    total     zero      percent
     contents   contents   entries   entries   zero
1    [illegible] [illegible] [illegible] [illegible] [illegible]
2    [illegible] [illegible] [illegible] [illegible] [illegible]
3    [illegible] [illegible] [illegible] [illegible] [illegible]
```

リスト3

```
GPSS sample list

BLOCK NO.   LOC      OPERATION     A,B,C,D,E,F,G
                     FUNCTION      [illegible]
                     GENERATE      [illegible]
                     TRANSFER      [illegible]
            [illegible] ASSIGN     [illegible]
            [illegible] QUEUE      [illegible]
                     SEIZE         [illegible]
                     DEPART        [illegible]
                     ADVANCE       [illegible]
                     RELEASE       [illegible]
                     TERMINATE     [illegible]
            [illegible] ASSIGN     [illegible]
                     GATE          [illegible]
                     GATE          [illegible]
                     TEST          [illegible]
                     TEST          [illegible]
            [illegible] ASSIGN     [illegible]
                     TRANSFER      [illegible]
            [illegible] GATE       [illegible]
            [illegible] ASSIGN     [illegible]
                     TRANSFER      [illegible]
                     START         [illegible]
                     END
```

　来月は目先を変えて遺伝的モデルを考えてみる予定である。できればはやりのファジーにも触れたいと思っているのだが。
　とはいえとも所詮「予定は未定」。すべては神の御心のまま。もっとも皆さんの意見によっては神様も考えを変えるかもしれないので、思うところがあれば、どんどん意見してほしい。
　ではまた、来月。

参考文献
・システムとシミュレーション　[illegible]　共立出版
・コンピュータシミュレーション　[illegible]　近代科学社

THE SENTINEL

■S-OSとゲーム

　システム中心の硬い記事が多いと思われがちなS-OSですが、初期の頃からゲーム開発パッケージBEMSを始めとして意欲的にゲーム分野にも取り組んできました。BEMSはアクションゲーム用のパッケージでしたが提言という色合いが強く、実際にアプリケーションゲームが発表されることがなかったのが惜しまれるところです。S-OSのゲーム史はアドベンチャーゲームやシューティングゲームをちりばめながらも、パズルゲームを中心として流れてきたといってもいいでしょう。
　開発の容易さから、TTCやSTACKなどで書かれたゲームの多かったなかで、今月は久し振りにアセンブラで書かれたアクションパズルゲームが登場します。しかもあのCOLUMNSとくれば、これは見逃す手はありません。X68000では付録ディスクで配布されたYETが有名ですが、S-OSユーザーとしては指をくわえて見ているしかありませんでした。タイルが降り積もっていく緊張感、次第に速くなるタイルの落下速度、連鎖反応を起こして一気に山が崩れるときの爽快感。どうぞ、このCOLUMNSで思う存分はまってください。

ブロックアクションゲームCOLUMNS

■COLUMNS登場

　このCOLUMNSは作者の堀井さんが初めて作った100行以上のプログラムだそうです。タイルの落下速度は次第に速くなっていき、ある程度速くなると最初の速度に戻って再び速くなっていきます。タイルの回転方向はゲームセンターのCOLUMNSやYETとは逆になっていますので、戸惑う方がいらっしゃるかもしれません。また、タイルのキャラクタが気に入らないという方もいらっしゃることでしょう。本文でも触れられているように、簡単に自分好みのキャラクタを設定できますので、改造してみてはいかがでしょうか。個人的には「ーーー　■」などが判別しやすくていいのではないかと思います。
　このプログラムは泉氏のX-BASIC講座実習で取り上げたYETのアルゴリズムを参考に作ったということです。OhXの賞活講座の連載ではさまざまなアルゴリズムが解説されていますので、自分でプログラムを作ってみるのに格好の教材となることでしょう。高級言語で書かれたプログラムをアセンブラに「移植」するのではなく、そのアルゴリズムを自分でアセンブラプログラムに直す。連載のこのような利用法も面白いのではないでしょうか。



■S-OSの系譜

　1987年4月号ではS-OSの歴史に新しい1ページが加わりました。初のアクションゲームが登場したのです。それまでS-OSのゲームといえばパズルゲームが主流でした。これには理由がありました。誰もがS-OSの文字表示は遅いと考えていたのです。
　S-OSは各機種のモニタやBIOSを利用して画面表示を行っています。しかも文字セットを統一するため、共通文字セットから各機種の文字セットへの変換を行ったあと、モニタ・BIOSを利用するのです。この2重の手間が介在したのでは、とてもアクションゲームを実現できるほどのスピードは望めない。スタッフを含め多くの人がそう思い込んでいたとしても不思議ではないでしょう。
　事実、E-MATEなどのエディタでも、文字表示用の各機種独自のサブルーチンを組み込んで使うことを前提としていました。もっとも、これはE-MATEが必ず1行80字の文字を出力している、つまり1画面2000文字のキャラクタをスクロールのたびに書き込んでいるという隠れた理由もあります。いずれにせよ、S-OSの表示が遅いという実例となってしまったのは間違いないでしょう。
　INVADER GAME、TANGERINEの2つのゲームは、これらの暗黙の思い込みを見事に打破してくれました。INVADER GAMEはその名のとおり古典的なインベーダーゲームですが、列をなしたインベーダーが下へ迫ってくる部分は結構重くなりそうな処理です。それに輪をかけて大変そうなのがTANGERINE。こちらは横スクロールシューティングゲームで、スクロールと自機・敵弾の重ね合わせという重そうな処理を抱えています。作者の片岡氏は、S-OSには難しいといわれているアクションゲームに敢えて挑戦してみたということです。ところが共通ルーチンしか使っていないのに、実際にはウェイトを入れなければゲームにならないほどの速度が出たとのこと。あえて不可能に挑戦してみるこの心意気が、新しいS-OS領域を生み出したのでした。

全機種共通S-OS"SWORD"要

ブロックアクションゲーム COLUMNS

"Beyond the columns" そして "Yet another columns"。COLUMNSの源流に忠実に。いまS-OS上にもCOLUMNSがやってきました（ゲーセン版とは関係ないですよ）。存分にプレイしてみてください。

Hori Toshikazu
堀井 俊和

S-OS "SWORD" 用の「COLUMNS」を発表します。このプログラムは1990年8月号で発表されたX68000用の「Yet Another Columns」の移植版です。X68000版を見て指がうずいていたS-OSユーザーの皆さんとうとうS-OSでCOLUMNSがプレイできるのです。

入力&実行方法

ダンプリストを打ち込まれる方は8000H～883EHをMACINTO Cなどで入力しその範囲でセーブしてください。実行アドレスは8000Hです。ソースで入力される方はエディタで打ち込んでREDAなどでアセンブルしてください。OFFSETがついているので、一度セーブしてロードしなおしてから実行してください。

ゲーム内容

もはや、多くの人が知っていると思いますが、念のため説明します。基本的な操作はテトリスなんかと同じです。上から落ちる3連のブロックを使って下に積もったブロックを消していきます。ブロックを消すには同じ文字のブロックを3個以上、縦、横、斜めに並べなければなりません。

キー操作は4、6で左右、5でブロックを入れ替える（3連ブロックの順序を下から上にずらす）。スペースキーでブロックを落とします。そのほか、Pでポーズ、シフトブレイクで中断です。テンキーのない機種では後述の変更例を参照してください。

得点は少々変わっており、落としたところ（内部的にはブロックが止まったところ）が下から何段目かを調べ、それから3を引き（この時点で1以下だとすべて2となります）それを2で割り、またそれに2をかけ、10倍したものがブロック1個の点数となります。これはプログラムの関係でそうなったことで別に意図はありません。

またスピードもゲーセン版テトリスのようになっておりどんどんスピードが上がっていって、最速を超えるとかなり遅くなってループを繰り返すというものです。

このゲーム、1周するのは簡単ですが2周目はかなり難しいでしょうね。だいたい1周を超えると下から1ブロックが積まれていったりしますからね。でもやっぱりハマりますよ。

プログラムについて

はっきりいって、私はこのゲーム、わりとすんなり完成しました。というのも、泉大介氏のX-BASIC入門をかなり参考にさせていただいたからです。このゲームのネックとなるブロックのチェックルーチンから、ブロックを消して落下させるという一連の動作の大まかなアルゴリズムを使わせていただきました。本当に素晴らしいアルゴリズムでした。泉さんありがとうございます。もうひとつ、乱数発生ルーチンをOh!MZ1986年11月号から使わせていただきました。プログラムの構成についてはソースにもある程度注釈を入れておきますのでそれを参考にしてください。また、チェックルーチンの内容は前述の泉氏の記事（1990年6、7月号）を見てください。

プログラムの変更

これは各機種で違うスピードの調節やキー入力の変更などを行うものです。ダンプリストから入力した方はすべて打ち込んでCRCなどを確認したあと、ソースリストから入力した方は該当部分を書き換えて再アセンブルしてください。

まず、キー配置の変更から

81D3H	左への移動
81D8H	右への移動
81DDH	ブロックの回転
81E2H	ブロックを落とす



に変更したい文字のキャラクタコードを書き込んでください。

スピード調節は何カ所にも分かれているのですが、ここではひとつだけ挙げておきます。あとは自由調節ということで。

速くするには80FFHと8104Hの2バイトを組み合わせた16進4桁の数値を小さくします（下限は0010Hまで）。逆に遅くするにはさきほどの2バイトを増やせばいいのです。この場合は0200Hくらいまでにしたほうがいいでしょう。

最後にブロックキャラクタの変更です。デフォルトではA、O、X、M、K、Sといった比較的変わった形のキャラクタを集めていますが、変えたいという方はご自由にどうぞ。8528Hからの6バイトに使いたいキャラクタを書き込んでください。

＊　＊　＊

このプログラムは私にとって初めての100行以上のプログラムです。S-OSと参考文献のおかげで挫折もせずに完成できました。まあ、それほどたいしたモノではないですがね。でもやっぱり完成すると嬉しかったものです。はっきりいって、あまり綺麗なプログラムではありませんが、プログラム本体やワークエリアにも注釈を付けておいたので参考にしてください。

また、このゲームは入門用の題材としてはとてもいいと思いますので、できれば皆さんも一から作ってみてください。テトリスもどきのほうが簡単だけど、こっちのほうが面白いんじゃないでしょうか。

リスト1

[illegible]

THE SENTINEL

```
[illegible]
```

▶X [illegible]turbo版グラディ[illegible]ウス[illegible]に感動してしまった。あんな素晴らしいものを作れる人が[illegible]同じユーザーとしてうらやましく思います。私も何年かかってもいいから、自分のやりた[illegible]いことをやってみようと心底思います [illegible]


```
[illegible]
```

全機種共通システムインデックス

- 序論 共通化の試み
- S-OS"MACE"
- Lisp-85インタプリタ
- チェックサムプログラム
- マシン語プログラム開発入門
- エディタアセンブラZEDA
- [illegible]グツールZAID
- ゲーム開発[illegible]BEMS
- [illegible]エフェクトレータ[illegible]
- インタラプト S-OS番外編
- マシン語入力ツールMACINTO-S
- Lisp-85入門 [illegible]
- 仮想マシンCAP-X85
- Lisp-85入門2
- Lisp-85入門3
- Prolog-85発表
- ポケコン・テーブルのお話
- FM音源ラウンドエディタ
- S-OS"SWORD"
- Prolog-85入門1
- magiFORTH発表
- Prolog-85入門2
- [illegible]ゲームJEWEL
- LIFE GAME
- 基礎からのmagiFORTH
- Prolog-85入門3
- スクリーンエディタE-MATE
- 実戦入門magiFORTH
- Z80TRACER
- magiFORTH TRACER
- ディスクダンプ&エディタ
- "SWORD" Z80 CG
- 対話で楽しむmagiFORTH
- PC-8801版S-OS"SWORD"
- FM音源ミュージックシステム
- FM音源[illegible]
- 計算カブ[illegible]のmagiFORTH
- SMC-777版 S-OS"SWORD"
- [illegible]
- MZ-2500版 S-OS"SWORD"
- FuzzyBASIC発表
- 明日に向かって magiFORTH
- ちょっと便利な[illegible]プログラム
- ディスクモニタ DREAM
- FuzzyBASIC料理法(1)
- [illegible]ゲーム HITTAN
- MAZE in MAZE
- FuzzyBASIC料理法(2)
- CASL & COMET
- FuzzyBASIC料理法(3)
- マシン語入力ツールMAC[illegible]
- FuzzyBASIC料理法(4)
- アドベンチャーゲーム MARMALADE
- テキスト作成ツール CONTEX
- [illegible]アニメがお好き
- アニメーション[illegible]MABE
- "SWORD"再構築とMAGICの標準化
- INVADER GAME
- TANKBRAIN
- S-OS"SWORD"変身セット
- MZ-[illegible]"SWORD"をQD対応[illegible]
- インタラプト コンパイラ特集
- FuzzyBASICコンパイラ
- エディタアセンブラZEDA [illegible]
- STORY MASTER
- パズルゲーム[illegible]
- [illegible]JACKWRITE
- FM[illegible] S-OS"SWORD"
- [illegible]逆アセンブラ[illegible]
- PC-[illegible] S-OS"SWORD"
- [illegible] CORE WARS
- FuzzyBASICコンパイラの[illegible]
- X1turbo[illegible] S-OS"SWORD"
- 序論 [illegible]のなかのマイクロコンピュータ
- 付録 S-OSの仲間たち
- もうひとつのFuzzyBASIC入門
- ファイルアロケータ&ローダ
- インタラプト S-OS[illegible]
- BACK GAMMON
- タートルグラフィックスパッケージTURTLE
- X1turbo[illegible]"SWORD"アフターケア
- [illegible]ルーチン
- PASOPIA[illegible] S-OS"SWORD"
- FuzzyBASICコンパイラ[illegible]
- [illegible]コンパイラ[illegible]
- シューティングゲーム ELFES
- [illegible]コンパイラ言語SLANG
- [illegible]ゲーム TRADE
- シミュレーションウォーゲーム WALRUS
- シューティングゲーム ELFES [illegible]
- [illegible]の作戦
- [illegible]言語SLANG入門(1)
- Lisp-[illegible]NAMPA シミュレーション
- マルチウィンドウドライバ [illegible]
- [illegible]SLANG入門2
- [illegible]エディタ WINER
- [illegible]エディタ TED-[illegible]
- アフターケア WINERの[illegible]
- SLANG[illegible]ライブラリ
- シューティングゲーム MANKAI
- シューティングゲーム ELFES [illegible]
- ノースジェネレータ SOURCERY
- パズルゲーム LASE LINE
- [illegible]ゲーム FLICK
- [illegible]エディタアセンブラ REBA
- [illegible] S-OS"SWORD"[illegible]
- Z80[illegible]ゲーム[illegible]
- SLANG[illegible]ライブラリ
- ノースジェネレータ RING
- [illegible]コンパイラTTC
- TTC[illegible]パズルゲーム TOBAN
- CP/M[illegible]ファイルコンバータ
- [illegible]シミュレーション BUGS
- [illegible]
- [illegible]パズルゲーム PUSH [illegible]
- SLANG[illegible]ディレクトリライブラリ
- DIO LIB
- SLANG[illegible]WORM RUN
- [illegible]SLANGコンパイラ
- [illegible]コンパイラTTC++
- [illegible]アセンブラ[illegible] Z80
- ファミコンエミュレーション[illegible]
- [illegible]STACK
- [illegible]フォーマット[illegible]
- STACK[illegible]ゲーム SQUASH!
- X68000対応S-OS"SWORD"
- PC-286[illegible]S-OS"SWORD"
- [illegible]アセンブラ[illegible]
- [illegible]
- BILLIARDS
- ライブラリアン WLB
- [illegible]エディタ EDC-T
- STACKコンパイラ

●[illegible] S-OS"MACE"または S-OS"SWORD"[illegible]しませんので[illegible]

▶[illegible]Oh!MZ[illegible]

愛読者プレゼント(Part2)

さてさて，先月に引き続き3周年記念プレゼントです。今月はソフトハウス編。
けっこうな量のプレゼントでしょ？　なんと今月だけで245名に当たるのだから，競争率は低くなるってもんだね。さあどれにするかか決まったらさっそくハガキを出してちょーだいな。
応募方法は168ページを見てね。

1

日本ファルコム　☎0425(27)6581



A.ドラゴンスレイヤーカードケース　10名
B.イースIIIカードケース　10名
C.リリアカードケース　10名
D.リリアスタンプ　10名
E.リリアバッジ　5名
F.リリアキーホルダー　5名
G.卓上カレンダー　30名
H.ファルコムバッジ　5名
I.ファルコムキーホルダー　5名
J.JDKバッジ　6名
K.JDKキーホルダー　5名

お馴染み日本ファルコムからは，こぉ～んなにたくさんのプレゼントグッズをいただきました。これだけで なんと100名分！ ファルコムファンは見逃せませんね。

2

ツァイト　☎03(299)6481



A. 3D倶楽部リビング
X68000用　5"2HD版　3,790円(税別)　1名
B. 3D倶楽部ダイニング
X68000用　5"2HD版　3,790円(税別)　1名
C. ねじ式
X68000用　5"2HD版4枚組　12,800円(税別)　1名

ツァイトからは，CGツールのデジタルクラフトのデータライブラリの3D倶楽部，そしてあのねじ式をプレゼント　各1名の方に。

3

東芝EMI　☎03(587)9005



A. ビデオ ヴァリスIII　3,500円(税込)　5名
B. CD FZ戦記 アクシス　2,500円(税込)　3名
C. CD AIRCOMBAT/IMPERIALFORCE/大戦略III'90　2,500円(税込)　3名
D. CD グラナダ　2,500円(税込)　3名
E. CD ラグーン　2,500円(税込)　3名

あの優子がなんとアニメーションになりました。このぶんだと，X68000版"ヴァリスIII"が登場するのもそう遠い…ことではないかな。CDのほうは最新のゲームミュージックをプレゼントです。

4

イマジニア　☎03(343)891

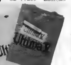


A.シムアースプロモーションビデオ　10名
B.シムアースポスター　10名
C.ポピュラスポスター　10名

はやくもイマジニアから発売が決定したシムアース，ゲームのほうはもうちょっと先だけど ビデオとポスターをプレゼントしちゃいます。ポピュラスファンの人のためにこちらのポスターも用意しました。

5

ポニーキャニオン　☎03(221)3161



A.ウルティマTシャツ　4名
B.ウルティマステッカー　30名
C.ウルティマシャープ&ボールペン　8名

ポニーキャニオンからは，あの有名なウルティマシリーズ，ウルティマVIのグッズをプレゼント。どれもほかではちょっと手に入らない代物です。

6

システムサコム　☎03(635)7609

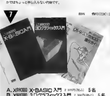


A.ジェミニウイングオリジナルブルゾン　1名
B.ジェミニウイング　X68000用　5"2HD版2枚組　8,800円(税別)　1名
C.アトミック・ロボキッド　X68000用　5"2HD版　8,800円(税別)　1名

ジェミニウイングの発売記念に作ったオリジナルブルゾンを1名の方に。そしてアーケードから移植されたバリバリのシューティング2種類を各1名に。

7

BNN　☎03(238)1323



A. X68000 X-BASIC入門　2,300円(税込)　5名
B. X68000 3Dグラフィックス入門　2,300円(税込)　5名
C. OS-9/X68000操作ガイド　2,300円(税込)　5名

X68000をゲームマシンとしてしか使ってない人ってわりといるはず。でも　これらで自作のゲームを作って遊んでいる人にはこれらの本は重宝するぞ。さあ，これであなたもプログラマ！

6

マイクロプローズジャパン　☎0423(33)774



A.GUNSHIP
X68000用　5"2HD版　14,800円(税別)　2名
B.GUNSHIP アパッチプラモデル　2名

本格フライトシミュレータとあって 人気の高いGUNSHIP。このゲームは攻撃用ヘリコプター「アパッチ」を操縦するってなわけで，今回はその「アパッチ」のプラモデルもあるのです。

9

T&E SOFT　☎052(773)7770

幻獣鬼

X68000用　5"2HD版3枚組
8,800円(税別)　2名



「遥かなるオーガスタ」ばかりが取りざたされているT&Eだけど，このゲームもなかなかのもの。転職システムや 豊富なアイテムなどでプレイヤーを飽きさせないアクションゲームです。

10

アートディンク　☎0474(77)7541

栄冠は君に

X68000用
5"2HD版3枚組
9,500円(税別)
3名



高校野球をシミュレートしたゲーム。バントや采配がやたらあるのもらしくていい。約4,000校の中から1校を選び 目指すはもちろん甲子園，そして全国大会優勝だ。深紅の旗が待っているぞ。

プレゼントの応募方法

とじ込みのアンケートはがきの該当項目をすべてご記入のうえ，希望するプレゼント番号をはがき右下のスペースにひとつ記入してお申し込みください(同番号内にいくつか種類がある場合は2－A，2－B　のようにアルファベットも明記のこと)。締め切りは1991年1月18日の消印分まで，当選者の発表は1991年3月号で行います。

11 ぴくせる君

M.N.M Software ☎0423(66)3064



X68000用　5″2HD版　4 [illegible]円(税込)　3名

もともとはＭＮＭの自社ゲーム開発用に作られたスプライトエディタ。機能はそんなに多くはないけれど，そのぶん低価格で使いやすいツールだ。

12 LUCY・SHOT

日本ソフテック ☎0425(82)1162



X68000用　5″2HD版2枚組　7 [illegible]円(税別)　3名

ソフテックのピンボールシリーズ第2弾。今回は前作の「PINBALL・PINBALL」とうってかわってアメリカン　コミック調。ちなみに背景の美少女の名前はルーシーです。

13 DISS-P

サザンエンタープライズ ☎03(787)3637



X68000用　5″2HD版　[illegible]円(税込)　2名

録音，再生，"編集が手軽に"できる多機能デジタルサウンドツール。録音再録はもちろん，編集もマウスでらくらくというスグレモノ。X68000にしゃべらせるスピーチ機能もついているぞ。

14 オリジナルトランプ

光栄 ☎045(561)6861



10名

信長の野望シリーズ最新作「信長の野望 武将風雲録」のイラストが印刷されたトランプ。おじさんがニッチをにらんでいるのがなんともコワイ。これで神経衰弱だけはしたくないと思う今日この頃。

15 オリジナルフロッピーケース

ザイン　ソフト ☎078(242)2856

10名

コンピュータを扱っていて，わりと不便だなと思うのがフロッピーの持ち運び。で，このかっこいいザインのフロッピーケースの登場だ。これを友達に自慢してやろう。くやしがるぞ。



11月号プレゼント当選者

[illegible]

以上の方々が当選されました。おめでとうございます。商品は順次発送いたしますが，入荷状況などにより遅れる場合もあります。また，公正取引委員会の指示により，このプレゼントに当選された方は，この号の他の懸賞には当選できない場合がありますのでご了承ください。

X68000 マシン語プログラミング 入門編

著■村田敏幸

[illegible]ページ　定価[illegible]円(税込み)

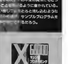
ストロングスタイルのマシン語入門!!

■本書登場のきっかけ

1990年11月現在，『Oh! X』誌に連載中の「X68000マシン語プログラミング」のうち＜入門編＞と称した部分を1冊にまとめたら，こんな本になった。マシン語プログラミングに興味をもったX68000ユーザーのための副読本，とでもいったらよいのだろうか。少なくとも，教科書的なプログラミング入門書では決してない。むしろ，問題集であり，実践テキストのブリに近い。

マシン語にかぎらず，プログラミングに関する知識/技術は，実際のプログラミングの中でこそ身につき，磨かれるものだ。この不変の真理にもとづき，本書は読者に自分の頭と体とを使うことを強いるように書かれている。エッセンスを100倍くらいに薄めて飲み下しやすくしたり，とろとろと流し込むような親切さは排除した。文書の糧に[illegible]，サンプルプログラムを読み，動かし，改造することによって[illegible]るだろう。

■本書の内容



- [illegible] 0 マシン語プログラミングの準備
- [illegible] 1 マシン語プログラミングの流れ
- [illegible] 2 68000の基本命令を覚えよう
- [illegible] 3 12種の68000実習プログラミング
- [illegible] 4 デバッガを使ってみよう
- [illegible] 5 文字列操作の基本
- [illegible] 6 正しいフィルタの作り方
- [illegible] 7 コマンド作成"基本"作法
- [illegible] 8 サブルーチンに汎用性を
- [illegible] 9 「プロセス操作」という世界
- [illegible] A ファイル管理の方法
- [illegible] B デバイスドライバを作る
- [illegible] C 続"入門編"のための[illegible]
- [illegible] [illegible] 版バージョンアップ[illegible]

'91年1月刊行予定

SX-WINDOW プログラミング

吉沢正敏■著

B5判変型　予価[illegible]円(税込み)

X68000にイベントドリブン方式のマルチタスク，ウィンドウ環境を提供するSX-WINDOWは，X68000に新たな世界を拓くものとして熱い期待を集めている。本書は，このSX-WINDOW上でプログラムを作ってみたいと思っているユーザーを対象にした，プログラム作成のためのガイドブックである。イベントドリブン，リソースなどのウィンドウ・プログラムの基礎知識，サンプルプログラムによる具体例，ウィンドウ環境のシステムコール一覧など，SX-WINDOW上でプログラミングする者のエッセンスを集めている。

バックナンバー案内

Oh!Xは1990年1月号から12月号までをご紹介しました。現在1989年10～12、1990年2～6、8～12月号までの在庫がございます。バックナンバーおよび定期購読のお申し込み方法については169ページを参照してください。

1990



1月号（品切れ）
特集1　オペレーティングスタイルの研究
特集2　Cプログラミング活用論
ショートプロばーでぃ/Z80's Bar
X68000マシン語/C調言語講座/DōGA CGA
●X1turbo用シミュレーションゲームSuper Battle
LIVE in '89　さよならを過ぎて/HYDEEN
THE SOFTOUCH　レナム/メタルサイト
全機種共通システム　WORM KUN/再帰型SLANG
特別付録　X68000 THE SOFTWARE CATALOGUE



2月号
特集　画像圧縮へのアプローチ
ショートプロばーでぃ/Z80's Bar/DōGA CGA
X68000マシン語/C調言語講座/X-BASIC講座実習
●X68000用ゲームプログラムGonGon
●MZ-700用画面Eyelarth
LIVE in '90　オーダイン/魔女の宅急便
THE SOFTOUCH　A-JAX/フラッピー2/夢幻戦士ヴァリスⅡ
マップ/トレット/Mu-1/CYBERNOTE PRO-68K
全機種共通システム　別1型コンパイラTTOI



3月号（品切れ）
特集　MUSICアドベンチャー
X68000用MIDIドライバ＆音源エディタ
なんでも鳴らせるOPMD X/MMLを簡易データに
ショートプロばーでぃ/Z80's Bar/DōGA CGA
C調言語講座/X-BASIC講座実習
●X1/turboシミュレーションCRISIS in Tokyo
LIVE in '90　パワードリフト/スチーム/となりのトトロ
THE SOFTOUCH　ナイトアームズ/新/ダンジョンマスター
全機種共通システム　多機能アセンブラOHM Z80



4月号（品切れ）
特集　ゲームシステム文学論
1989年度GAME OF THE YEAR発表
ショートプロばーでぃ/Z80's Bar/DōGA CGA
X-BASIC講座実習/C調言語講座/X68000マシン語
●X1 MZ-2000/2500用RPG The Cave of Doll
●うわさのX68000ついに登場
LIVE in '90　イーニングフォース（OPMD対応）
THE SOFTOUCH　The Fille Professor/HOST PRO-68K
全機種共通システム　ファディエンピューターシミュレーターMY



5月号（品切れ）
特集　BASICプログラミング
第5回　言わせてくれなくちゃだワ
ショートプロばーでぃ/Z80's Bar
X-BASIC講座実習/X68000マシン語プログラミング
●新機種X68000SUPER HD/EXPERT II/PRO II
●ワンコンスティックの製作
LIVE in '90　TURBO OUTRUN
THE SOFTOUCH　天下統一/ポピュラス/Hyperword
全機種共通システム　インタプリタ言語STACK



6月号
特集　創刊8周年記念PRO-68K（付録5"2HD）
Oh!Xアンケート結果大分析大会
ショートプロばーでぃ/Z80's Bar/PurePASCAL
X-BASIC講座実習/X68000マシン語プログラミング
●X1turbo用コマンドシェルシミュレータ
●ハードウェア工作入門
LIVE in '90　ナイトアームズ/悪魔城伝説/この木なんの木
THE SOFTOUCH　三國志II/FAR SIDE MOON/グラナダ
全機種共通システム　X68000用S-OS SWORD



7月号（品切れ）
特集　マシン語への第一歩
X68000SUPER HD試用レポート
ショートプロばーでぃ/Z80's Bar/DōGA CGA
X-BASIC講座実習/PurePASCAL
●INTEGRAL X―ハードウェアへの挑戦
●ハードウェア工作入門
LIVE in '90　夢幻戦士ヴァリスⅢ/ドッカーラとファーガー演習
THE SOFTOUCH　サークIII/Lu/ダウンタウン熱血物語
全機種共通システム　月刊ケーブルアセンブラWZD



8月号
特集　ADVANCED 2D GRAPHICS
100号記念特別モニタプレゼント
ショートプロばーでぃ/Z80's Bar/INTEGRAL X
X-BASIC講座実習/X68000マシン語プログラミング
PurePASCAL/ハードウェア工作入門
●X68000用画像回転プログラム　XROT II X
LIVE in '90　OMENS OF LOVE/ENDLESS RAIN/ダーティフォックス
THE SOFTOUCH　大航海時代/ウルティマV/プロミストランド
全機種共通システム　リンカWLK



9月号
特集1　日本語を処理するための序論
特集2　ADVANCED 2D GRAPHICS
ショートプロばーでぃ/Z80's Bar/DōGA CGA
X-BASIC講座実習/マシン語プログラミング
PurePASCAL/ハードウェア工作入門
●青木和人流プログラミング道場
LIVE in '90　風の谷のナウシカ/ラジオ体操第一
THE SOFTOUCH　1917 Again/シムシティー/ギャラガ'88
全機種共通システム　BILLIARDS



10月号
特集　電子音楽理入門
ショートプロばーでぃ/Z80's Bar/DōGA CGA
マシン語プログラミング/ハードウェア工作入門
青木和人流プログラミング道場
●気分屋の大人のためのX68000
●中級者のようこそC言語へ
LIVE in '90　Rise And Fall/PARADOX/キューピー3分クッキング
THE SOFTOUCH　ワールドコート/ルーンワース/闇の血族/遥かなる記憶
全機種共通システム　ライブラリアンWLB



11月号
特集　理科系のGAME REVIEW
Z80's Bar/DōGA CGA/カードゲーム
マシン語プログラミング/ハードウェア工作入門
PurePASCAL/X-BASIC講座実習
ようこそC言語へ/INTEGRAL X1
気分屋の大人のためのX68000
LIVE in '90　ピラミッド/ノーサリアン/ザ・スキーム
THE SOFTOUCH SPECIAL　シナリオ対談/ナイトロポンの対談
全機種共通システム　スクリーンエディタEDC T



12月号
特集　XCのための傾向と対策
X-BASICプログラミング講座/ハードウェア工作入門
マシン語プログラミング/ショートプロばーでぃ/Z80's Bar
大人のためのX68000/ようこそC言語へ/INTEGRAL X1
シミュレーションプログラミング入門
●特別企画アナログジョイスティックの製作
LIVE in '90　グラディウスIII/メタルサイト
THE SOFTOUCH SPECIAL　イメージファイト/ソーサリアン対談
全機種共通システム　STACKコンパイラ

［第8話］

パソコン戦線異状なし

TAKAHARA HIDEKI 高原 秀己

珍しくパソコンの話。

ぼくの周りを見ると、最近になって、新顔乗しくパソコンを仕事場で使いはじめた人が何人かいる。

そう、ブック型とかノート型という新顔ラップトップ機を使っているのだ。

これまではこわごわワープロを清書機代わりに使っていただけの人が

「パソコンはやっぱりOSを使えなきゃあ、ね」

とした顔をして解説しながら、一太郎などを使って悦に入っているのを見ると、バカバカしいのを通り越して微笑ましくもある。

彼らはいずれも今年になって、PC-9801n（98ノート）とかPC-286noteFを買ったクチである。

さて笑ってないで、自分のことを考えてみると。

とんでもないことに気がついてしまった。

ぼくはこの1年、パソコン（X68000とかじゃなく、ごくごくありふれた機械）をなにひとつパワーアップしていないのである。例外的に「エコロジーII」と「ノートン・ユーティリティ」を買ってユーティリティを活用することになったのを除けば、ソフトもハードも含めてまったく変化のない1年であった。

もちろん、その他のエレクトロニクス製品にしたところで、スーパーファミコンを買ったわけでもなく、BSもレーザーディスクもまたまた買わずじまいだった。

だが、スーパーファミコンは注文する意欲がなかっただけだし、BSとレーザーディスクはレンタルビデオですませた、という大義名分があるからいい。いずれにしても、なければないでいいものだったから、さして問題ではない。

ところがパソコンを1年間何もシステムアップしなかったのは、まだパソコンが黎明期を脱したにすぎない時代であることを考えると、実に問題だ。

もしも技術革新が激しかったのに、ぼくが何も追従しようとしなかったとすれば、これは衰退を意味する。革新への追随を止めるということは、研究所が新しい研究をせず、工場が新たな設備投資を止め、そしてスーパーや外食企業が新たな店舗を開拓しなかったことと同じなのだ。

ぼくは新しい技術に目を向けなかったのだろうか？

そのことを確認するために、パソコン関連ビジネスの今年の出来事を列記してみると――。

ノート戦争が本格スタート。

－ビジネス用で386高性能、i486マシンが続々登場。

－LAN管理ソフトをめぐりメーカー/ソフトハウスで派閥分かれる。

ハードウェアがますます大容量化。

PCエンジンのカードソフトがそのまま使えるポータブル版が発表。この後、ベールを脱いだスーパーファミコン。

4MビットダイナミックRAMが量産開始！。

こんなところだろうか。

このうちゲーム機はジャンルが異なるのでオミット。4Mはまだパソコン向けに普及する段階ではないのでパス。ハードディスクの技術的大容量化はこれまでの延長なのでパス。ましてやLAN管理ソフトは新会社ができただけなので無視、とはしていくと、実に冒頭の光景として紹介したノート型パソコンの普及といった程度しか残らない。この他で記憶に残っていることといえば、エプソンがリンドバーグをCMに使ったり、PC電子手帳が出たり、と実にくだらないことばかり。

実際のところ、革新的なことは何もなかったようだ。WINDOWSは機能のオマケにはなったが使わなくてはならない状態にはいたっていないし、日本語環境を変える新しいフロントエンドプロセッサが登場したわけでもない。画期的なプリンタが登場したわけでもなく、メモリカードも普及しなかった。

ソフトにしても「ナントカ2」や「アイカワラズ3」といったパート2ものばかりで、画期的なブームを呼んだゲームが出たわけでもなく、革新的なビジネスツールもお目にかからずじまい。

そう！　間違いとして、ぼくが悪かったわけではなく、マーケットにあまりにも変化がなかったのだ。

改めて考えてみると、今年ぼくは「なにか新しくて役に立つものはないかな」と何度秋葉原に行ったことか。よく行く新宿のパソコンショップでビックリするようなゲームやプリンタを探したが、一度として見つかったことはなかったのだ。

かくいうぼくの使い方は、パソコン通信用の端末がほとんどで、あとはワープロ、各種分析用の123（統計書ソフト）、ゲームといった、きわめて固定的なシンプルなものだけに、「なにか」というのがなかなか難しいことはわかる。

それでも革新的なものさえあれば、50万円規模の投資は考えていただけに、ちょっと残念だ。

そもそも、ノート型パソコンの普及のみが目立ったということは、PC-9801ならびにエプソン製互換機がまたまた前進したことを示しているだけだ。NECを追う富士通だのシャープだのが巻き返しをする気配はカラキシない。

あまりに何もなさすぎたのだが、このままの状態が続くようであれば、来年もヒマつぶしに大幅値引きになっているノート機を買って終わり、という状態になるのがオチだろう。

だが本当はそんなことではいけないはずだと思う。まだまだパソコンは常に新しいシーンをぼくたちに見せ続けてくれるべき時期を脱していないはずだ。

変に安定成長を決め込んで、成長を拒否するようでは先が見えてしまう。刺激のないマーケットは退廃を生むだけ。退廃すれば、それで終わりだ。

1991年に期待。

ジョブズはやっぱり天才だ！

Arita Takaya 有田 隆也

スーパースターの来日

たしかにこう感じました。「スティーブ・ジョブズはやっぱり天才だ！」。いまもまだ輝きの残るAppleII、毎日お世話になっているMacintosh、そして新機種発表で華々しく突入したNeXT。この魅力あふれる3台のマシンを世に送り出してくれたジョブズが来日し、1時間半にわたる圧倒的なプレゼンテーションを見せてくれました。10月1日から5日まで開かれていた情報処理学会30周年記念国際会議"infoJapan"の最終日の目玉として彼の招待講演が催されたのです。

会場である京王プラザホテルのホール入り口前では開場前から多くの人が待っていて、これから始まろうとするジョブズの講演に対する期待が充満していました。なるべく前に座りたいと思い、早くから待っていたところ（前から4列めぐらいにもぐりこめた）、すっと横をあの人なつっこい顔が通りすぎていきました。ずいぶんと背が高いな、びっくりすると同時にそう感じました。ふてぶてしいまでの余裕にあふれた態度は相変わらずといったところでしょうか。

ジョブズは講演前日の夕方、40人（あるいはfourteenかも）のスタッフとともに来日したそうです。そして、スタッフたちは徹夜で会場設営のセッティング、ジョブズ本人も朝8時からリハーサルを行ったという話も漏れ、期待も高まる一方といったところです。

さらにうれしいことには、発表されたばかりの新しいNeXTも持ってきていて、デモンストレーションが行われたのです。今回の講演は情報処理学会という格式高い学会の主催であり、タイトルも「90年代のコンピューティング」ということで、新しいNeXTの宣伝が前面に出ることはないにしても、やはり、新しいNeXT。そう、「カラーネクスト」に対する期待はいやがおうにも高まります。

圧倒的なプレゼンテーション

ジョブズは長身ながらステージの上を軽く動き回ります。そして、時々観客に問いを投げかけます。「どうですか？」という軽いものから、「これこれがわかる人いますか？」という質問まで。彼はパーソナルコンピュータの発展史における革命を順番にたどりながら進んでいきます。ジョブズらしく、ソフトウェアにおける革命のみを対象としています。

まず第1の革命がスプレッドシートです。それを担ったマシンとしてIBM PCが引き合いに出されます。そして、会場の向かって右側のほうにセッティングされていた新しいNeXTにインストールされている"Improv"というソフトの紹介（画面は正面の特大スクリーンに投影されます）。あまりピンとはきませんでしたが、このImprovは新しいNeXTのウリのひとつらしいのです。ジョブズの持つ「1990年代のコンピューティング」の姿を少しずつ明らかにしながら、実質的には新しいNeXTの宣伝をやっているようなものでした。

第2の革命はデスクトップパブリッシングです。これを担ったマシンとしてMacintoshが挙げられます。この説明では特にNeXTを使ったデモンストレーションはありませんでしたので、ちょっと気になりました。疑問に思った人が質問したのに対しては別に意味はないと答えていました。新しいNeXTではワープロ関係では特に目新しいバージョンアップがなかったのでしょう。

第3の革命はカスタムソフトです。はっきりとした説明はなかったのですが、ソフトウェア作成環境などをさしているようです。マシンとしてはSUNワークステーションが挙げられます。ここでは、NeXTのインタフェイス作成支援ソフトウェアであるインタフェイスビルダのデモンストレーションがありました。最近、NeXTをちょっといじっているので特に驚きませんでしたが、少しバージョンアップしているようです。

いよいよ第4の革命。これこそがNeXTが背負って立つ、今後5年間のキーワードということです。それは"interpersonal Computing"です。個人個人を結びつけるグループ全体を考えたコンピューティングが極めて重要であるということを、ジョブズは強く主張します。考えてみれば、1980年代のパーソナルコンピューティングは文字どおり個人個人の知的領域を拡張するものでした。そして、それが拡大すればするほど、ほかの人の領域との交わりも広がってきます。そのための、自然かつ緊密な融合こそが"Interpersonal Computing"の目的だということです。別の見方をすれば、計算機はメディアとしての第1歩を踏み出していくということなのでしょう。

新しいNeXTを使って、この第4の革命の一端が示されます。拡張された電子メール機能の紹介が主でした。単なるテキストから始まり、ボイスメール、チェロ奏者ヨーヨーマの演奏、計算機プログラム、それから、これも今度のNeXTのウリらしいのですが、FAXも自由自在に転送してみせました。内蔵のOCRソフトが、FAXイメージのテキスト領域をASCIIテキストに変換してくれるそうです（モデムはオプション）。

圧巻のカラー実演

このようにして、講演は4つの革命を追って淡々と進んでいきます。4つめの革命"interpersonal Computing"を担うのはNeXTなのだと強くアピールして、一応ひとつの節は終わります。しかし、ここから聴衆の待ち望んでいたことが始まりました。いうまでもなく、新しいNeXTのカラー機能の実演です。

新しく用意されたカラー機能にもいろいろバリエーション（選択肢）がありますが、真のカラー（True Color）を実現するには32ビットカラーボードを接続する必要があります。このボードにはインテルのi860（33MHz）とJPEG形式のデータ圧縮プロセッサが接続されています。これによりリアルタイムにビデオを取り込み圧縮したり、解凍して表示することができます（1.4Gバイトのハードディスクを接続すればなんと1時間分といわれています）。

ジョブズはステージ右のほうに設置された新しいNeXTを説明しながら操作し、観客は正面の大スクリーンを見つめます。なにかの映画の一部がNeXTの上で再現されます。カラーの絵を加工するソフトはなかなかのものでした。スポーツカーの窓ガラスごしに見える景色がリアルに見えるときには思わず観客席からため息が洩れていました（マウスでスポーツカーをつかんで高速に動かしたりもしました）。

フィナーレは息を飲むほどあざやかでした。カメラでジョブズ自身の上半身をリアルタイムに取り込みます。手のひらを上に向けたところを取り込んだかと思うと、ドナルドダックの絵を切り取ってきて、それをその手のひらの上に載せたのでした。どっと拍手が湧きました。500人ほどの聴衆たちはことばさえ発せずに、ただただ拍手をしつづけていました。

アメリカ人の質問

講演後は自由に質問を受けつけ、それにジョブズが答えるというものでした。残念ながら、日本人はほとんど質問をせずアメリカ人ばかりでしたが、そのうちのいくつかを簡単に紹介しましょう。

＊　＊　＊

Q　UNIXで標準的になりつつあるウィンドウシステム、X WINDOWについて。

A　X WINDOWはMacintosh以下である。NeXTをベストなUNIXマシンと見る人がいるが、別にUNIXマシンを作るのが目的ではない。1%にすぎないUNIXユーザーのためのマシンを作ったのではないのだから。

＊　＊　＊

Q　CISC対RISCについてどう思うか？

A　速いRISCができたのならば、それはすばらしいと思うし、もっと速いCISCができたのならば、それはよかったと思う。どんなに速いプロセッサができたかということより、速いプロセッサを使ってどのようなマシンを作るかということにこそ、意味があるのだから。

＊　＊　＊

ジョブズのこの2つの答えは、誰もが使えるマシンにするということ（ヒューマンインタフェイス指向）ということをマシンにやらせるかということ（ソフトウェア指向）をストレートに表しており、Macintosh、AppleIIなどを生む思想の本質的な部分に関わる考え方であるということができると思われます。ところで、ジョブズに対してある人がこう質問しました。

＊　＊　＊

Q　Virtual Realityについてあなたはどう思いますか？

＊　＊　＊

それに対してジョブズは具体的なことはなにも言いませんでしたが、でも、大きな興味を持っているというようなことだけは言っていました。

仮想現実とは

Virtual Reality、日本語に直せば「仮想現実」です。最近、この概念はちょっとしたブームになっているようで、特集を組んでいる雑誌もあります。この概念は、現実ではないのに現実のように感じさせる技術のことを表しています。というとすぐに、脳に直接信号を入れるようなSF的な話を連想してしまいますが、そこまでいくような話はいまのところないようです。

人間を入出力をもつ装置と考えるとき、理論的には、入力のみ、つまり五感に対して正確な情報を与えれば、それが実在しない仮想的なものでも人は現実だと思い込んでしまいます。たとえば、コンタクトレンズ状のもので正確に映像が映るものを目につけ、鼻と耳と舌にも気がつかないくらい小型でしかも高性能な匂いと音の発生器と味覚刺激器をつけておきます。それで、それらを統一した場面を構成するように刺激を同時に与えて再現してやれば、人もだまされるだろうというわけです。

でも、触覚は難しいものがあるでしょう。人が出力をも同時にもった装置であることから発生する別の難しい問題と絡むからです。たとえば、手でものをつかむというごく当たり前の出力（行動）を考えてみましょう。腕の回りにセンサでもつけて腕の行動を検知します。腕を動かすと同時につけたコンタクトレンズ型表示器のなかの腕も動かしてやります。ここで、ほかの物体をつかむような状況を仮想的に実現させようとするときには難しいことになります。実物体につかもうとした物体をうまく表示するだけでなくて、手のひらの適当な位置にものがあたったという感触を与えてやらなければならないからです。この制御をリアルタイムでやるのは大変な問題でしょう。

この仮想現実はいまのところごく限られたことにしか応用されていないようですが、ヒューマンインタフェイスとしては究極的なものであることには間違いがないでしょう。とんでもない応用が無限に考えられるような気がしませんか？



とびっきりの美女との関係

最近、NeXTを自由に使うことができるようになって、暇を見つけてはいじってます（といっても古いNeXTですが）。気に入った点、素晴らしい点はいくらでも挙げますが、いやな点はひとつだけです。それは有名なことかもしれませんが、やたらと遅いことです。もともと、とても静かなマシンなので、待たされているときでもハードディスクを読んでいるからか、プロセッサがなにか計算でもしているからか、わからなくて、暴走したのかなと一瞬不安になることもよくあるほどです。

しかし、今度のNeXTはCPUが68030から68040に変更されたということもあり、この一大欠点はかなり解消されたのではないでしょうか。

ジョブズによると、古いNeXTのもつ1）値段が高い、2）速度が遅い、3）ソフトが少ない、4）カラーが出ない、という問題点を新機種はすべて解決したとのことです。4）は確かでしょう。3）はどうだかはっきりはわかりません。そして、1）と4）の問題は同時には解決しているような気もしますが、総合的には見事なマシンを発表してくれたなという気がします。

ジョブズの顔に絶えず浮かんでいる笑みとあふれるような自信。生まれつき、あるいは、噂にきくとびっきりの美女と結婚したばかりとのことで、そのことも手伝っているのかもしれません。しかし、僕にはAppleII、Macintosh、と生んできた彼がいよいよ真打ちを生み終えたと実感している表情に思えてなりませんでした。

第55回

猫とコンピュータ
過激なCRTと共に

Takazawa Kyoko
高沢 恭子

前回に引き続き「第7回ホビーマイコンショウ」の模様をお届けします。これを書き上げるのにも並々ならぬ苦労があったようで――。でも，猫の話題にはついつい目がいくところがキョウコさんらしいですよね。

前回からのつづき，「ホビーマイコンショウ」の会場で，パソコン通信についての取材を受ける話を書こうとしたとき「洗濯物をとりこむ猫」をテレビで見てしまった。

アニメーションの世界なら，猫が家事をやろうが，楔出をやろうが，夢は広がるばかりかもしれないが，これは家庭用ビデオで撮影した実写だったので，なんだか困ったような気持ちになった。

白黒模様のその猫は，共稼ぎの中年のご夫婦に飼われていて，婦人の留守宅で洗濯物をとりこんでいたそうだ。はじめは誰がやったのかわからず不気味だったが，あるとき現場を見て知った。人に言っても信じてもらえなかったので，苦心してビデオにおさめたらしい。

マンションのベランダで，猫はハンガーの洗濯物をひとつずつ口でひっぱってはずしては，部屋に運びこんでいた。少し不鮮明な画面がいかにもドキュメンタリーらしくて，映像の説得力はやはりすごいなと思った。

少し困ったなと感じたのは，人のいないところでは，イタズラか悪いことをやる習性のあるのが猫だったのに，かくれた善行らしいことをやっている猫がいて，証拠ビデオつきで公表されてしまったことだ。

でも，人間とかけひきをするのがじょうずな猫が，ほめてくれる人もいないところで「お手伝い」なんてムダなことをするだろうか。これは猫の哲学に反する。

とっても可愛がられていたこの猫は，たぶん主人夫婦の帰りを待つ孤独をくりかえして大きくなった。夜になると，帰宅した奥さんはまず洗濯物をとりこみ，それからこの猫にゴハンを与える。猫の頭の中では，洗濯物のとりこみとゴハンは，ワンセットの連鎖的な行事になった。

夕方からだんだん夜になり，寂しいしおなかもすいてくる。ふと奥さんの帰宅したときのようすが猫の脳に思い浮かぶ。そうだ，洗濯物だ。ベランダの洗濯物をとりいれると，つぎはゴハンだ。猫は洗濯物を部屋に入れることで，連想的に奥さんの帰りと晩ゴハンを実現できるような期待を持ったのだろう。

「拾われて可愛がられたことへの恩返しだと思う」というご主人の言葉にあえてさからって，猫はじぶんのためだけに行動すると主張したい。

火を吹くCRT

と，ここから前回のつづきを書こうとして，キーをひとつ叩いたとたん，突然CRTは真っ暗になり，パチパチというおそろしい音をあげはじめた。

ブラウン管の中で火花が散りつづけ，いまにも破裂する気配も感じられたので，これからどうなるのかを見届けたい気持ちを押さえて電源をきった。

もちろんテキスト部分をセーブするゆとりもなくスイッチをきったので，ここまでの文章は思いだしながらの，2度目の入力になった。セーブのアクションをしていたら，パソコン本体は損傷していたかもしれない。

CRTを本体から分離して，もう一度電源を入れてみた。こんどは「ピーッ」という大きな悲鳴をあげて，緑色の火花が背面全面であざやかに光った。

テレビが古くなったり，ホコリがたまって発火した話は聞いたことがあるけれど，このCRTはまだ4年とちょっとしか使っていない。部屋全体もそれぞれのマシンの周囲も，できるだけ清潔に保つことを心がけてきたつもりなのに，こんなマンガのような現実がわが家でおこるなんて。

驚いたりあきれたりしながら，つくづくCRTをながめた。ともかくメインになっていちばん働いてくれたCRTだったが，そんなにひどい扱いをしていたのかな。ほとんど1日つけっぱなしという日もあるけれど，OA機器はそれがふつうだ。いったい中身はどんなようすなのか，ついでに背中のフタをはずしてのぞいてみた。

ホコリはとくに目立たない。最終確認のため電源をつないでみる。いちばんうしろの部分の細い線の束から，激しい火花と音が出た。ただのショートかもしれないが，いよいよ爆破寸前の様相になってきたので現場保存にとどめた。夫はいつも，帰宅のたびにコワイ事件が待っているので，疲れるだろうな。

本体をX68000のCRTにつないで，うまいぐあいにエディタのしごとは継続できた。

通信鬼ケ島

やっと，前回からのお話。

FBI NETのメンバー，編集マンの取締さんからの紹介で，某女性週刊誌の記者氏とカメラマン氏があらわれた。場所はアキハバラのラジオ会館8階大ホール「第7回ホビーマイコンショウ」開催中の会場。

取材の目的は，女性ネットワーカーたちそれぞれの，パソコン通信とのかかわりかたについてらしい。取材される面々は，FBIの女性メンバー6人。うち3人は20代のお嬢さんたちで，この1年くらいのあいだに加入した新メンバーだ。

パソコン通信の効用が，このところいちだんと言われるようになってきた。すでに大きな役割をはたしている商用データベースは別格として，会社内で，大学や研究所で，お役所，公共の施設，サービス機関などで，つぎつぎに新しいネットワークの手段として利用されはじめている。

情報の共有，選択の任意性，保存，転送などの点で，同じパーソナルな通信でも，電話やファックスにはない活用法を，たくさん持っているといえる。

取材に訪れた週刊誌を発行している大手S社は，パソコン誌も出していて，社内ではいくつもの雑誌がそれぞれに通信ネットを持っているそうだ。この日の「女性S」誌も「Sネット」という通信局をおいて，読者との交流をはかっているが，ほんの少数がアクセスしているだけとのこと。パソコン通信の話題が高まってきたこの機会に，もう1歩踏み込んで，理解と普及につとめてみようというつもりらしい。

まずはひとりずつの写真撮影。ショウの展示に使っていたマシンの前で，6人が順番にキーボードを叩くしぐさをしてみせる。パソコンの取材では，このワンパターンのポーズはどうしても覚悟しなければならない。カメラマン氏がしごとを終えて先に帰ると，ホールの奥にある別室で，取材となった。

「取締」さんからは，取材記者氏がパソコンとは無縁の方ということを聞いていたが，それはとても良い点がある半面，お互いになかなかの苦労があるだろうと予測できた。「パソコン通信」は，パソコンを使って通信をするだけのことであっても，マシンの面と，ネットワークとしての面の双方から，成立ち，しくみ，特徴や特異性までもふくめて，全容を説明するのはたいへんなことだ。ふと，2年くらい前に，いまよりもっとパソ通が未開のころ，趣味の話をする座談会で「通信」からの代表で出席して，とてもむずかしかった記憶がよみがえった。

「鋼鉄」さんも同席して，若い記者O氏の質問がはじまる。パソコン通信をはじめた動機，チャンス，あるいは目的，そしてじっさいの利益や楽しみはなんなのかといったことが，端から順に問われていくのだが，用語の1つひとつに注釈が必要なので，なかなか難行が続く。

ホスト局，ホストプログラム，端末プログラム，シスオペ，アクセス，ID，パスワード，メニュー，ボード，階層構造，アーティクル，オンライン，アップロード，ダウンロード……などなど，そして通信族の特殊用語や表記術，OFF会の存在や交際の流儀まで，とキリがない。

それを説明しているのが「ちやこ」さんに「えいりあん」さっちゃん，「ちゃーりー」さん，「exit.」さん，「鳩」さんといった，お互い本名もよく知らないハンドルネームの一団だ。真剣にいぶかしそうに質問をつづけるO記者が，鬼ヶ島を訪れた桃太郎みたいに見えてきた。



パブリックをめざして

先日NHKテレビの「婦人百科」で「絵本づくり」を指導しているのを見て，なんとなくパソコン通信を連想した。

手づくり絵本のしごとは，オリジナルな絵とお話をつくることが半分で，もう半分はその手で製本することだった。わが子や孫へ心をこめた贈りものをしたいという動機から，愛好者も多いらしい。

もう何冊か手づくりの絵本を制作した人たちの話では，もちろん読み聞かせたい絵とお話があっての「絵本」ではあるけれど，ハードの分野ともいえる製本にかける喜びはとても大きいようだ。このあたりが現状でのパソコン通信にとてもよく似ていると感じた。

パソコン通信も，目的はネットワークづくりとコミュニケーションであるはずだ。でもいまはコミュニケーションの内容よりパソコンという手段の新しさや有能さが注目を浴びている。

キーボードから叩き出した情報が，モデムの仲立ちで電話回線に乗せられ，遠く離れたパソコンに送られるのだ。もともと，情報の移動や，編集，整理分類，収納が得意なパソコンが，距離と空間を越えてその力を発揮できるようになったのだ。これは注目されるだけのことはある。

だからうっかりすると，こんなふうに機械だけで取り上げて「さあ通信をはじめましょう」と呼びかける一方で，ますます高度で有能な「パソコン通信」が開発されて，じっさいには手軽さから遠ざかっていってしまう心配もある。

パソコン通信は，男女を問わず，なんの用意もない人が取りかかろうとしたら，そんなに簡単にはいかない。意欲も知識も根気もいる。費用もかかる。いちばんたいせつな「目的」を持ちつづけられなければ，ムダに終わる。

女性週刊誌だから女性を取材するのはしぜんなことだけれど，独力ではじめる女の人は少ないかもしれない。ここにいるメンバーも，じつは好都合な環境に負うところが大きい。「ちやこ」さんは新聞社のワープロオペレータ，「えいりあん」さっちゃんはシスオペ夫人，「ちゃーりー」さんはプログラマー，「exit.」さんは東京につとめる電気のプロ，「鳩」さんは通信機器メーカーの社員だ。

みんなじょうずに質問に答えて，パソコン通信の楽しさを語っている。良い記事に仕上がって，読者の共感を呼ぶものになればいいと思う半面，結果的に不親切な報道をまたひとつつくることにならなければいいなと感じたひとときだった。

＊　＊　＊

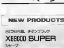
ところで，キーを叩いている椅子のそばに，背中が黒くコゲついたCRT，PC-KD851が死んでいる。やっぱり新しいCRTをひとつ買わなくてはダメかしら。X68000といちいちつなぎかえていたのでは，能率もカッコも悪いものね。

PENGUIN INFORMATION CORNER
ペ・ン・ギ・ン・情・報・コ・ー・ナ・ー

NEW PRODUCTS

SCSI内蔵、チタンブラック
X68000 SUPER
シャープ

X68000 SUPER



シャープは、「X68000 SUPER-HD」のハードディスクなしのモデル「X68000 SUPER」(CZ-604C-TN)、およびチタンブラックの専用ディスプレイ「CZ-606D-TN」を発売した。

「X68000 SUPER」は「X68000 SUPER-HD」と同じようにSCSIインタフェイスを標準装備しているので、先頃発売された光磁気ディスクなどのSCSI規格の周辺機器を接続できる。

また、標準ではハードディスクは内蔵していないが、追って発売される増設用3.5インチハードディスク「CZ-68H」(80Mバイト/アクセスタイム19ms)が内蔵可能である。

価格は「CZ-604C-TN」が348,000円、「CZ-606D-TN」が79,800円(ともに税別)。「CZ-68H」(平成3年2月発売予定)は価格未定。

〈問い合わせ先〉
シャープ㈱ ☎03(260)1161、06(621)1221

「書院」シリーズ、新4機種
WD-A321/341/630/730
シャープ

シャープは、毛筆体も内蔵した液晶表示ラップトップモデル「WD-A324/341」と「トリプルフォント&マルチアウトラインフォント」搭載のCRTモデル「WD-A630/730」の合計4機種を発売した。

「WD-A324/341」は「書院スーパーアウトラインフォント」に新たに毛筆体も内蔵し、名刺に使えるような小さな文字から、晴れ幕やポスターに使えるような大きな文字まで104種類の文字サイズをアウトラインフォントで印刷できる。価格は178,000円と198,000円(ともに税別)。

「WD-A630/730」は明朝体、ゴシック体(ROM)、毛筆体(正楷書体、FD)のトリプルフォントを標準で装備。さらに、毛筆体(行書体)や教科書体の別売フォントを加えたマルチフォントの拡大文字(4倍角以上の文字)すべてに対し、アウトラインフォント機能が自動的に働くようになっている。価格は145,000円と178,000円(ともに税別)。

〈問い合わせ先〉
シャープ㈱ ☎03(260)1161、06(621)1221



WD-A341



WD-A730

ハンディイメージスキャナ
ファインスキャナ-X68
HAL研究所

ファインスキャナX68



HAL研究所はX68000専用のハンディイメージスキャナ「ファインスキャナX68」を発売する。

「ハンディスキャナX68」はハンディスキャナでは最高クラスのグレイスケール256階調取り込みを実現している。読み取り幅は105mm。専用インタフェイスボードによる高速転送で20mm/sの移動速度での読み取りが可能。

専用ソフト「IMAGE PHOTO 68」では取り込み画像をメモリ上に持ったまま画面を見ながら調整することが可能で、従来の明るさやコントラストなどを変えながら何度も取り込みを行う手間を解消している。

また、調整項目のγ値などは細かい項目がユーザー指定できる。特にユーザー自編集による調整は強力。

画面出力モードに関わらず、メモリ上に256階調データを持っているので、拡大/縮小操作を行っても画面の美しさは損なわれない。このため、画面の解像度とプリンタの印字密度の違いを十分に吸収できる。

そのほかZIMをはじめ多彩なファイル形式をサポートしている。

価格は未定だが本体、インタフェイスボード、IMAGE PHOTO 68込みで4万円程度に抑えられる模様。モノクロながら高品位な画面が得られるツールとして期待される。

〈問い合わせ先〉
㈱HAL研究所 ☎03(252)5561

アナログRGBをS端子に
XAV-1S
電波新聞社

電波新聞社はアナログRGB信号の入力をビデオあるいはS端子信号に変換して出力できるアダプタを12月末に発売する。

入力は21ピンRGBマルチ、15ピンアナログ15KHz(200ライン、標準解像度)に。出力はS端子、ビデオ信号に対応していて、外部ACアダプタによる電源供給が必要。

パソコンなどのアナログRGB出力はもちろん、キャプテンシステムターミナルの出力や文字放送のRGB出力をS端子に変換するなどの用途も考えられる。価格未定。

〈問い合わせ先〉
電波新聞社 ☎03(445)6111



XAV-1S

TVゲーム用バトルシート
MENKURI
アイアンクラフト

アイアンクラフトは、ジョイスティックを固定することのできるTVゲーム用シート「MENKURI」を12月21日から発売する。

この「MENKURI」は市販されているジョイスティックを固定してプレイすることにより、操作性、操作フィーリングを向上させようというもの。ジョイスティックホルダー部分は前後に80mmスライドするので使用者の体型や好みに応じてセッティングできる。また、脚部にはプラスチック製のアジャスタがついていて、安定したプレイが可能。

対応する機種は、以下のとおり。
アスキースティックXターボ(アスキー)
サイバースティック(シャープ)
XE-1シリーズ(APは不可)(電波新聞社)

価格は7,800円(税込)+1,000円(送料)で、パソコン誌などを媒体とする通信販売のみで取り扱われる。

〈問い合わせ先〉
アイアンクラフト ☎0256(33)6111

MENKURI



INFORMATION

第5回 ファンタジー&ロールプレイングゲームフェア

大阪の旭屋書店では毎年恒例の「ファンタジー&ロールプレイングゲームフェア」を90年12月1日から91年1月15日までの期間、開催する。

○商品 ファンタジーシミュレーション、テーブルトークRPG、カードゲーム、メタルフィギュア、ダイス、専門誌バックナンバー、関連図書、コミック、ゲームブック以上の販売。および、テーブルトーク、カードゲーム全商品1割引き「ファンタジーRPGイラスト」の募集、展示などの特別企画が予定されている。

〈問い合わせ先〉
旭屋書店 ☎06(353)4011

イマジニア協賛による
杉本彩コンサート

ザビュウス、シムシティなどのヒットで今年はおおいにパソコンゲーム界を賑わせてくれたイマジニア。そのイマジニアが「ファンサービスの一環として」杉本彩のコンサートの協賛をする。

○タイトル みんなゴーシャス!
○場所 渋谷クラブ・クワトロ



○日時 1月7日(日) 開演 19:00

イマジニアでは、このコンサートにペア75組、150名を招待。応募方法は官製ハガキに、
1) 名前
2) 住所
3) 電話番号
4) 雑誌名(Oh!X)
を明記したうえ、
〒163 東京都新宿区西新宿2-7-1新宿第一生命ビル13階 イマジニア株式会社
杉本彩コンサート係
まで。平成2年12月25日必着。

〈問い合わせ先〉
イマジニア㈱ ☎03(343)8911

BOOK

OS-9/68000 マルチユーザーシステムガイド

本書は「RAINBOW OS-9ガイド」に続くOS-9ガイドブック第2弾であり、OS-9/68000の最大の機能であるTSS(Time Sharing System)によるマルチユーザー機能を解説したものである。X68000版完全対応。価格は3,300円(税込)。

〈問い合わせ先〉
スピリットパブリッシング ☎0258(92)4810



このインデックスは，タイトル　注記——筆者名，誌名，月号，ページで構成されています。いよいよ1991年に突入，あけましておめでとう。さて受験生の人，あともう少しです。コタツで眠っちゃダメですよ。

参考文献
I/O　工学社
ASCII　アスキー
コンプティーク　角川書店
テクノポリス　徳間書店
ポケコンジャーナル　工学社
POPCOM　小学館
マイコン　電波新聞社
マイコンBASIC Magazine　電波新聞社
LOGIN　アスキー

一般

▶ネットワーク　ホットライン　第30回
「ほのまの　てわなしみの[illegible]　いがらしみきお氏作『パソ通験所体験ゲームBBSちゃん』を紹介　PDSはPC-9801用ホストプログラム「owns」「TURBO BBS」。——編集部，LOGIN，21号　272-273pp.
▶ネットワーカーズホットライン　第31回
オンラインゲーム特集。ネットワークを使って　他のアクセス者と楽しめるネットワークマルチプレイヤーゲームを紹介。——編集部，LOGIN，22号　272-279pp.
▶レノワトライ　コンピュータミュージック
主にPC-9801についてコンピュータミュージックの始め方や製品紹介を行っているが，X68000も含めたソフトのデータ互換・規格やMIDIの規格についての記事も掲載されている。——塚本・山田拳一　マイコン　2月号　21-154pp.
▶キーボードのなくなる日
最近注目度が高まってきている手書き入力の現状について　Macintosh，MS-WINDOWS，myXTなどを例題に見ていく——編集部，マイコン　12月号　72-177pp.
▶[illegible]のゲーム[illegible]
AMショーリポートの巻。各社大型きょう体の新作ゲームが目立ったほか　ガンシューティングのゲームやテーブルゲームにも魅力ある作品が出品されていたとのこと——[illegible]　マイコン　12月号　236-237pp.
▶やまさんのアルゴリズム　ブック
最終回。ノスト[illegible]の一環として　ハイアワノ一[illegible]の考え方を理解しプログラムを作る——やまさん　マイコン　12月号　254-258pp.
▶ビジネスマンの情報活用術
シャープから発売された「パイ」電子システム手帳PA-9500について特徴と機能　表示[illegible]などさまざまな面から紹介——[illegible]　マイコン　2月号　323-326pp.
▶[illegible]ハード入門
ガイガーカウンターの原理と動作　今回は部品入手などの事情から秋月電子のキットを組み立てる　放射線と[illegible]する[illegible]記事つき——石川[illegible]，マイコン　12月号　338-381pp.
▶第2回サイクロンCG大会
9月24日に行われた第2回サイクロンCG大会の模様と[illegible]作品の紹介をしている。——編集部，ASCII　12月号　440p.
▶シャープ　パイ電子システム手帳 PB-Z
シャープの新製品電子手帳PA-9500は　タッチパネルの採用　[illegible]　手書き入力など今までの電子手帳にない[illegible]を[illegible]している　その[illegible]とこれからの[illegible]開していて述べる。——[illegible]　ポケコンジャーナル　12月号　4-9pp.
▶プログラムBASICカードエミュレータ BE
電子手帳のプログラムBASICカードのプログラムを支援するソフトウエア"BE"の構成とクロス環境の方法についての解説。——[illegible]　ポケコンジャーナル　2月号　13-24pp.
▶なんでもQ&A
AltとVoltaのモデムについて「ノコン[illegible]を行う」は　自動着信の解除方法は　などの質問に答える——編集部　マイコン　12月号　420-427pp.

MZシリーズ

MZ-1500(BASIC MZ-5Z001)
▶ミューラア
キャは魔道師ミューラアを倒せるか？　地面をよけつつ　敵を倒し脱出したりするという　アクションゲーム——FROG　マイコンBASIC Magazine，12月号　124-126pp.
MZ-2500(BASIC-M25)
▶ヘンテコヘンテコMAZE
見知らぬ地域の中の迷路を通って　つぎの部の入り口へいく。全7面のへんてこりんパズルゲーム——[illegible]部，マイコンBASIC Magazine，12月号　127-178pp.

X1/turbo/Z

X1シリーズ
▶TESLA
降ってくるカラーブロック　色を合わせてぶっ壊せ！——久保田，マイコンBASIC Magazine，12月号　154-155pp.
▶LAST MISSION
ズオを6方向に移動させ　敵弾を振り切って倒しダメージを与える　しかし面ごとに違った命令があり　それに従わなくてはならないのが「ノ」。——竹井利明ほか，マイコンBASIC Magazine　12月号　156-158pp.
X1+FM音源ボード（FM音源ドライバ）
▶MGA版「ロディウス ～エンディング～」
ゲームミュージックプログラム——[illegible]　マイコンBASIC Magazine，12月号　[illegible]-169pp.
X1turboシリーズ
▶NEW SOFT
[illegible]ゲーム　プール　オブ　レイディアンスを紹介——編集部，LOGIN，2号　27p.
▶Cyber Mission
第2弾スクロールシューティング——伊藤敬之，マイコンBASIC Magazine　2月号　159-[illegible]pp.

X68000

▶NEW SOFT
アトミック　ロボキッド　イルーサの遺産　ダイナマイト　デュークほか　グラフィックツールのCANVAS PRO-68Kを紹介——編集部　LOGIN　21号　[illegible]-31pp.
▶[illegible]ゲーム徹底解剖
ラグーン　ナイアス　サイバリオン　ジェミーウイングを徹底攻略——編集部，LOGIN，21号　144-179pp.
▶X68000新聞
ニューラル　ギア　シュヴァルツシルト　イメージファイト　C compiler PRO-68K ver2.0のその後，新作情報はダイナマイト　デューク　銀河英雄伝説II，Magical Shot。——編集部，LOGIN，2号　252-255pp.
▶NEW SOFT
ノル　フィース　生中継68の紹介。——編集部　LOGIN　22号　25-29pp.
▶X68000新聞
シャープ社の新製品「XBASIC CHECKER」「SCSIボード」「光磁気ディスクユニット＆ディスクカートリッジ」を紹介　新着ゲームは「新ダンジョン マスター」「ダイナマイト　デューク」「銀河英雄伝説II」「日本語入力フロントエンドプロセッサ　FIXER Ver 4.5」——編集部，LOGIN　22号　252-255pp.
▶[illegible]ゲーム[illegible]
アクションRPG「ラグーン」を攻略——編集部，LOGIN，22号　90-153pp.
▶MAGIRY FOR X68K
究極！　FM，ADPCM，MIDIに対応したミュージックドライバ。コマンドリストとプログラム本体を掲載。——永田英哉＆あんど　マイコンBASIC Magazine，2月号　61-53pp.
▶Spell of Vision III
画面中央の自分を[illegible]で[illegible]ゴールを目指す。——[illegible]，マイコンBASIC Magazine，12月号　152-[illegible]pp.
▶[illegible]
うまく間合いを生かし　勝ち抜いていき　最の大将を倒す。ジョイスティック専用——林[illegible]　マイコンBASIC Magazine，12月号　54-165pp.
▶ROO LAND ～ 面エンディングBGM～
ソーサリアンのゲームミュージックプログラム——M.H.，マイコンBASIC Magazine，12月号　90-191pp.
▶トーニングフォース ～Snow Land～
前号のマイコンBASIC Magazineで発表されたMAGIRY用のミュージックデータ　トーニングフォースのBGM，[illegible]MT32[illegible]曲——永田英哉，マイコンBASIC Magazine，12月号　90-124pp.
▶MS-DOS入門ブック
MS-DOSの入門用[illegible]が　コマンドリファレンスなどHuman68kを扱うX68000ユーザーにも参考になる——編集部，マイコンBASIC Magazine，12月号　別冊付録
▶チャレンジ!! アドベンチャー　ゲーム
闇の血族　上巻を，画面写真で紹介——佐久間[illegible]　マイコンBASIC Magazine　2月号　225-227pp.
▶SOFT RADAR
新ダンジョン　マスター　ピンボール　ピンボールを紹介——編集部，POPCOM，12月号　10-11，2pp.
▶ゲームがオレを呼んでいる！
ソーサリアンのレビューとラグーンの攻略法，ナイアスの魅力など。——[illegible]　たかはび　アビゲイル金万　POPCOM，12月号　78-85pp.
▶WE ARE THE X68000 WORLD
アトミック　ロボキッド　闇の血族　完結編，ダイナマイト・デューク　パロディウスだ！　生中継68，ニューラル　ギアなどの新着　開発中ゲームと　光磁気ディスク　SCSIインタフェースボードを紹介——編集部，POPCOM，12月号　50-53pp.
▶ミュージック・ピアノ
「パラダイスの楽園（JAJA）」のミュージックプログラム——[illegible]　POPCOM，12月号　167-[illegible]pp.
▶GAMING WORLD
ナイアス　アクシス～FZ戦記～　Magical Shot，イルーサの遺産　闇の血族　完結編　ニューラル　ギア　ダイナマイト　デュークを紹介——編集部，テクノポリス　2月号　16-33pp.
▶SOFT EXPRESS
新ダンジョン　マスター　ピンボール　ピンボールを紹介——編集部，コンプティーク　12月号　51-37pp.
▶X68000 SPIRITS
ラグーンの攻略法，遥かなるオーガスタ　シュヴァルツシルト　ダイナマイト　デュークを紹介。——編集部　コンプティーク　12月号　252-258pp.
▶ゲームレビュー
期待度抜群　ジボール新サッカー級とナイアス　アクシス～FZ戦記～の批評記事——WAKKOH　あゆさわかつみ　[illegible]　マイコン　2月号　225-235pp.
▶なんでもQ&A
付属ワープロで[illegible]の文字列を削除しておく方法，TYPEコマンドでプリンタのティアオフ機能を使うには　ビデオレコーダーとX68000の接続について　など。——編集部，マイコン　2月号　424-435pp.
▶AXプログラミング講座
[illegible]のスプライトを[illegible]するテクニックをスプライト[illegible]を[illegible]説明する。——[illegible]，ASCII，12月号　36-368pp（リスト481-422pp）
▶AV STRASSE
X68000用SCSIボードと光磁気ディスクユニットの診断機能　CANVAS PRO-68Kの批評記事が掲載。——[illegible]，ASCII　2月号　409-412pp.
▶FREE SOFTWARE INDEX
ここ数ヵ月の間に主要ネットにアップロードされたPDSの一覧　X68000用のフロッピー高速化プログラム　アニメーションツールなどの高機能も掲載されている——編集部　ASCII，12月号　453-454pp.
▶[illegible]ロードテスト
シリーズ第4回，X68000の[illegible]について　PDSなどの充実と　BGMを鳴らしながらほかの作業ができるなどの機能に感心している——編集部，ASCII　12月号　468-479pp.
▶GAME BOX
闇の血族，ラグーン　熱血高校ドッジボール部サッカー編の紹介　批評記事——[illegible]　I/O　2月号　130-155pp.
▶PIX.C
以前掲載された画像ローダのバージョンアップ版。アセンブリ言語で記述してスピードアップされ　オプション機能も充実——Zenobia，I/O，12月号　186-187pp.
▶XBAS III C CHECKER
シャープから発売されたXBAS III C CHECKERについて　記述エラーの具体例や動作テストを通してその性能を探る——[illegible]　I/O，12月号　190-[illegible]pp.

ポケコン

PC-E500
▶とってもDIFFICULT
オセロに似たパズルゲーム　とっても難しいぞぉ——[illegible]，マイコンBASIC Magazine，12月号　170p.
▶Varunas
惑星システムのコンピュータが暴走　我が基地が[illegible]された[illegible]のシューティングゲーム——[illegible]　マイコンBASIC Magazine，12月号　171-[illegible]pp.
PC-E500/E650/1460/1401(6)
▶4R T
敵をよけながら時間内に[illegible]を数えるというゲーム——オッさんズ[illegible]　ポケコンジャーナル　12月号　67-69pp.
▶2LINE GAME
リストがわずか2行というゲーム2題。——せとけん，ポケコンジャーナル　12月号　20-71pp.
▶スザンNG
2枚組を2セット作る「6種マージャンゲーム」を女の子が[illegible]ようにした[illegible]。——せとけん　ポケコンジャーナル　2月号　72-74pp.

新刊書案内



年末が近づくと　そこかしこの出版社から今年の用語集的とかトレンディとかキーワードがどうとかいう本が出版される。今回はそんな本のなかから　電波新聞社のエレクトロニクスキーワード集を選んでみた。1項目につきキーワード＋関連事項という構成が非常に読みやすかったらが……真面目に読む科学技術の本ではなく　眠く気にぺらぺら読めるのが一番正しい読み方。今の世の中ーナ兜はファジーと呼ばれるほど　わけのわからないカタカナ用語が溢れかえっている。年末くらい……ういった軽い読み物でカタカナをチェックしてみるのもいいだろう。[illegible]組み合わせた　Macintoshの[illegible]にEMSが入っているという不思議な話や「ワープロの進化はパソコンにとってはひとつの脅威である」といった開拓けな記述（今はパソコンの[illegible]がワープロにとって脅威なのだ）もあるが　おおむね楽しんで1985年に注目された技術や古くからあってもまだ注目する技術がわかる。
巻末にある付録の各種データもなかなか興味深くて面白い。 [K]
エレクトロニクスキーワード集　1985年度版電波新聞社編　電波新聞社刊　☎03(445)6111　A5判　232ページ　1,800円



東京都市学校2　バッドシティの独職学
東京都市学校の2数目は　表現活動（マスコミ[illegible]を含む）をしている人が[illegible]エッセイで語られる。エッセイというより　東京都市学校でなされたレクチャーをまとめたものというほうが正しい。バッドシティというのは東京は未来都市であるが　かつて予想された未来とは似ても似つかぬ混沌としたバッドな未来都市という意味。林海象，[illegible]　コンプレなど興味れは多彩。 [K]
[illegible]　TOTO出版刊　☎03(3[illegible])[illegible]　A5判　285ページ　1,700円



柔らかい機械　思考のメカニズムの追求
人間の精神はどういうメカニズムになっているか　という厄介な問題に取り組んだ本。コンピュータの進化に伴って　人間の精神に対するアプローチも変わってきた。その基本にあるのが　従来のシリアル・コンピュータから新しいコネクショニスト・ネットワークという流れである。このコネクショニスト・ネットワークを中心に　脳や精神を探る。やや難解だが　先月紹介した「心の社会」とあわせて読むと面白いだろう [K]
ジェレミー・キャンベル著　中島健訳　青土社刊　☎03(263)7620　四六判　422ページ　2,700円

Q X-BASICを使っていてエラーを出すとエラーの内容とその行を出力してくれるので便利なのですが，そのあとでLISTなどとして画面を消してしまうとその行がどこであったのかがわからなくなってしまうんですね。またRUNしてそのエラー行までたどり着くのに時間がかかる場合もあるのでどうにかしてRUNさせずにエラー行を知りたいのですが。なにか方法はあるのでしょうか？

千葉県　[illegible]

A エラー行をLISTさせる方法はもちろんあります。実をいうとX-BASICにはマニュアルには載っていないが使える機能というのがあるようです。ここではX-BASICのマニュアルでは説明されていないLISTの使い方についていくつか紹介しましょう。

X-BASICのマニュアルを見てみるとLISTの使い方は

書式　[l] list [<行番号1>] [－[<行番号2>]]

省略形　[l] l.

となっています。

問題はこの行番号の部分です。まず，多くのBASICではそうなのですが，X-BASICでも行番号を省略するのに

.（ピリオド）

を使うことができるのです。この“.”は現在の行を表すものです。

たとえばプログラムをRUNさせて100行でブレイクキーを押して止めたのであれば“.”は100を表しますし，LOADでプログラムを読み終わったのであれば“.”はプログラムの一番最後の行を表します。

したがってプログラム実行中にエラーが出た場合も“.”は中断した行を表しているわけですから，これを使ってエラー行を表示させたいのであれば

l.（list .）

でエラー行を表示させることができます。またエラー行から後の表示やエラー行までの表示もBASICの書式に従って

l.－

l.－

とすればできます。

また，もうひとつLISTコマンドの隠された機能に，“ある関数の始まる行からのリストをとる”というのがあります。つまりfunc～endfuncの“func”のある行から表示するやり方です。これは

l 関数名－

で使うことができます。

たとえば，

1880 func drawmaze()

1890 /* 迷路を描く

2120 endfunc

となっているのであれば

l drawmaze

で

l. 1880－

と同じ働きをします。何度もリナンバーを繰り返しているときなどは便利に使えるでしょう。ただし，l －関数名という使い方はできませんので注意してください。

（吉村　[illegible]）

Q X68000付属のワープロで半角文字の混ざった単語を登録しようとすると，範囲指定がされているのに「範囲指定をして下さい」となってしまって，単語を登録することができません。X68000では半角文字を辞書に登録することができないのでしょうか？

秋田県　[illegible]

A おっしゃるとおり，付属のワープロで半角文字を使い登録することはできません。私も決まって半角で書く文字なんか（たとえばX68000の68000の部分とか），いちいち手作業で半角に変換していました。

ワープロを使っている途中で単語を登録するんだったらまだしも，わざわざワープロを起動して単語を登録する人はいないと思います。だってDICM.Xがあるからです。そうなんです，DICM.Xなら半角文字だって登録することができるのです。きっと[illegible]さんは，辞書ASKをワープロでしか使っていないとか，そういった人なのでしょう。DICM.Xは登録する単語の品詞情報の数がワープロに比べて豊富なので，副詞や連体詞を登録するときはこちらを利用することをすすめます。

おっとDICM.Xを知らない人，システムディスクのBINディレクトリの中にありますから用意してください。使用方法は付属のワードプロセッサ，辞書ユーティリティユーザーズマニュアルの，辞書ディスク保守ユーティリティに詳しい説明がありますから使用してください。

Q X-BASICのRND文の乱数のシードを初期化するっつうのがよくわかりません。

宮城県　[illegible]　大次

A まずは次のプログラムを実行してみてください（X-BASIC上で）

10 for i=0 to 10

20 　print rnd()

30 next

見たとおり10個の乱数を表示するプログラムです。Okと表示されていますね。それではもう1回実行してみてください。どうです，なにか気づきませんか。そうですね，先ほどの実行結果とまったく同じ乱数が表示されていますね。実は乱数といっても出目は決まっているのです。上のプログラムは1000回実行しても，ずーっと同じ乱数しか出ないのです。これでは乱数とはいえませんよね。

これは乱数が数式で計算されているのが原因です。コンピュータで使う乱数（疑似乱数）は一定の範囲の数字ができるだけデタラメな順番で出てくるように調整されています。その数字の並びには限界がありますので，何度も何度も乱数を使うと出てくる数字はいずれ最初と同じになってしまいます。この数列内部の並び方を変えるものが乱数系列と呼ばれるものです。

これを変更するのがRANDOMIZE文で，そのとき与える引数が乱数のシード（種）と呼ばれるものです。X-BASICでは－32768≦x≦32767の範囲で指定できるようになっています。シードを初期化するということは乱数の出目を変えるということなのです。

RANDOMIZE文が含まれていないプログラムは自動的に

RANDOMIZE(－32768)

が指定されたことになるので，毎回同じ実行結果になってしまうのです。

では，毎回異なる乱数を出すにはどうしたらいいでしょうか。これはRANDOMIZE文の引数に内部タイマの秒や分やカレンダーを使う手法が有名です。たとえば

5　RANDOMIZE(VAL(RIGHT$(TIME$,2))*100)

を先ほどのプログラムに追加して同じように2回実行してみてください。実行結果が変わっているでしょう。

リスト1

```
*************** sample.s ***************
* sample.s
        .include    doscall.mac

        .text
        .even

sample:
        pea.l   mes1
        DOS     _PRINT
        addq.l  #4,sp

        sub.l   [illegible]
        lea.l   [illegible]
        move.l  a1,-(sp)
        move.l  a0,-(sp)
        DOS     _SETBLOCK           * メモリブロックの変更
        addq.l  #8,sp

*****************************
*    子プロセスの起動
*****************************
        clr.l   -(sp)           * 自分の環境ポインタを指定
        pea.l   p1              * コマンドライン格納領域
        pea.l   fil             * コマンド行のポインタ
        move.w  #2,-(sp)        * モード2
        DOS     _EXEC
        lea.l   14(sp),sp
        tst.l   d0
        bmi     error           * エラー

*       ここからがモード0

        clr.l   -(sp)           * 自分の環境ポインタを指定
        pea.l   p1              * コマンドライン格納領域
        pea.l   fil             * コマンド行のポインタ
        clr.w   -(sp)           * モード0
        DOS     _EXEC           * 子プロセスの起動
        lea.l   14(sp),sp
        tst.l   d0
        bmi     error
*****************************
*    子プロセスの終わり
*****************************

        pea.l   mes2
        DOS     _PRINT
        addq.l  #4,sp
        DOS     _EXIT           * プロセス終了

error:
        pea.l   er_mes
        DOS     _PRINT          * エラー表示
        addq.l  #4,sp
        DOS     _EXIT           * 異常終了

        .data

mes1:
        dc.b    '子プロセスを起動します',13,10,0
mes2:
        dc.b    '子プロセスを終了しました',13,10,0
er_mes:
        dc.b    '子プロセスの起動に失敗しました',13,10,0
fil:
        dc.b    'ED.X',0
        ds.b    [illegible]
p1:
        ds.b    [illegible]
p2:
        dc.b    0

        .end
```

ない？変わっていない？追加した行は，内部タイマの秒に100をかけたものをシードとしています。だからシードは0≦S≦6000です。実行結果が同じなのは，あなたが1秒以内に2回プログラムを実行したためにシードの値が変わらなかったからです。さすがに3回実行すれば結果も変わるでしょう。これでは60種類の乱数パターンしかできないので，さらに複雑な計算式でシードを算出してやればいいでしょう。ちなみにシードを100倍しているのはシードの値が近いと出てくる乱数も似てくる傾向があるからです。

そのほか，シードを初期化する以外に

REPEAT:RND():UNTIL INKEY$(0)<>""

のようにして乱数を取り出す位置を変える手もあります。コンピュータを使うとどうしてもデタラメな値というものが作りにくいので，人間がキーボードを叩く時間間隔で乱数を調整しているわけです。

Q DOSコールを使って自分で作ったプログラムを子プロセスで起動したいのですが，うまく動きません。プログラマーズマニュアルを見ながらいろいろとやってみたのですが，それでも駄目なのです。どうやればいいのでしょうか。

鹿児島県　沢木　明夫

A アセンブラから子プロセスを起動するにはDOSコール$FF4B，EXECを使います。プログラマーズマニュアルを見ると，5つのモードがEXECに用意されています。このうち，子プロセスの起動にはモード0と，モード2の2つだけ使います。

まず最初にモード2を使います。モード2は指定した実行ファイルを環境で設定したパスを検索して，子プロセス起動のための情報をワークに設定します。ですからカレントにない実行ファイルも，パスが通っていれば大丈夫です。

こうしてからモード0を使います。モード0はワークに設定された情報から実行ファイルをロードして実行まで行います。気がついたかと思いますが，自分でワークに必要な情報を設定しておけばモード2を無理に使う必要はありません。

たったこれだけの手順で子プロセスを起動できるのです。たぶん沢木さんが子プロセスを起動できなかったのはメモリブロックの変更をしてなかったからだと思います。つまり，親プロセスに最大メモリが割り当てられるため，子プロセスを読み込むメモリがまったくない状態だったのです。

最後に子プロセスとしてED.Xを起動させるプログラムを紹介します。参考にしてください。

（[illegible]）

質問にお答えします

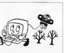
日ごろ疑問に思っていること，どんなことでも結構です，どんどんお便りください。編集，寄稿，編集室が総力を上げてお答えいたします。ただし，お寄せいただいているものの中には，マニュアルを読めばすぐに回答が得られるようなものも多々あります。使用前，マニュアルは熟読しておきましょう。質問はなるべく具体的に機種名，システム構成，必要なら図も入れてこと細かに書いてください。また，返信用切手同封の質問をよく受けますが，原則として，質問には本誌上でお答えすることになっていますのでご了承ください。なお，質問の内容について，編集部から問い合わせることもありますので電話番号も明記してくださいね。

宛先　〒100 東京都[illegible]区[illegible]2-10-13 NS[illegible]ビル

ソフトバンク株式会社出版部

「Oh! X質問箱」係

STUDIO X ON AIR



FROM READERS TO THE EDITOR

いよいよ新しい年、1991年がやってきました。皆さん、去年はどんな年だったでしょうか。よい年だった人たちは、今年もよい年でありますように。また、あまりよい年ではなかった人たちは、今年こそよい年でありますように。

◆1月号は発売日の冬休みに近くの本屋で買いました。テスト期間中ということもあって、みんながノートのコピーをやっているようだったけど、気のせいでしょう。教室で何気なく読んでいて「理科系のGAME REVIEW」のところをめくったとき、近くにいた友達が「難しい本読んでるねー」と言いました。それを聞いて「あっそうか、ぼくってえらいのか」と思わず心の中でつぶやいてしまいました。2月頃、為人にクラスチェンジしてしまいそうで、恐いです。
臼井 玉三郎(16)愛知県
「理科系のGAME REVIEW」を読んだんだから、数学と理科は大丈夫。てなぐあいにはいきませんよね

◆現在、私と友達の神二紀とでシミュレーションゲームを作っています。三国志に似た"出水伝"というゲームです。神二紀は絵がとても上手で、主に登場キャラクターの顔を描き、私はゲームのシステムを考えてプログラムを作ります。私たちの理想はとても高いので、なかなか紙上の段階から抜け出せません。どんなゲームかと申しますと、地形はうちの近所(半径5kmぐらい)でありまして、武将として登場するのはみんなうちの中学校の卒業生です。作画を開始してから、はや一年が過ぎましたが、これはとても笑えます。完成はいつになるか、とても予測はできませんが、うちわでそうやって楽しくのもとても面白いものであります。
堀川 健一(17)岐阜県
知っている人しか面白くないだろうけど、それでもいいですよね。作るのを楽しんでいるんだから

◆X68000を買う前、ほかのゲーム雑誌を見ると『ああ、X68000ってなんでもできるんだなあ』と思い、Oh!Xを読むと「ああ、X68000ってなんにもできないんだなあ」(ちょっとオーバー)と思った。そして、1カ月半前、X68000を買ってわかった。X68000はなんでもできる。できないのは自分だ、と。でも、ずっと夢に見ていたX68000を手に入れられたのだから、すごくうれしい。
鹿部 直人(17)静岡県
早くあなたのX68000がなんでもできるようになるといいですね。

◆みんな悲しんだもの、なにかためらってしまうよ。受験なんて言葉、数年ぶりなんだろう。2、3カ月前に一通ハガキ出したんだけど、そのとき年をごまかしちゃった。ごめんネー。同一女性だっていう、とがアピール点かな。ところで吉本ばななの「満月の夜」っていう詩集読みましたか? 彼女って絶一線的な表現するし、きっと天才だって思う。いま私にとって一番気になる存在なのよネ。この詩集にはその天才ぶりがあからさまに出ているのです。恥ずかしながら1回同じ目指している私には頭をばりばり食いた感じで「ばなな」以上のショックでした。
矢吹 幸子(25)福島県
同一星的というのは、森高さんにぴったりの言葉のような気がしますね

◆7月、X68000を買った。10月、原付(ホンダDIO SR、161,000円)を買った。もう僕には金も残っていない。「パロディウスだ!」が出る。ほしい。金がない。欲しい。バイトしなくっちゃ。時間がなくなる。X68000が使えない。勉強もしなくっちゃ。ギターの練習もしたいし。ガォー すいません。つい叫んでしましました
岩崎 貴和(2)富山県
落ち着いてください

◆8月21日、初めて両親と乗って高松へ行きました。以前のJR四国の車両はつまらなかったのですが、瀬戸大橋ができてからは本当にしゃれたカラフルでいろいろな列車が乗り入れてきて鉄道ファンにはうれしいかぎりです。ところで、高松へ行った理由はその際で結果は見事でした。しかし、Oh!Xをしっかり読んでいれば必ず受かると自信がつきました。4月も行くぞ。
瀬川 賢光(32)香川県
その際というのは、あの情報処理試験でしょうか。何級を受けたのかはわかりませんけど、高松までわざわざ行くんだから、今度こそは受かるといいですね。

◆日本シリーズが始まる。巨人有利の説が多く聞こえてくるが、巨人ファンの私は大胆にも4勝2敗で巨人の日本一を予想するのであった。もし、予想が外れたら編集部みんなで私を大笑いしてください。
新川 浩一(2)静岡県
(大笑)

◆秋の夜長、皆様いかがお過ごしでしょうか。やっぱり、ゲームにどっぷりつかっている人がほとんどでしょう。かくいう私も最近ほとんど睡眠不足状態が続いていて、体力が限界……。こんなとき、パソコンのように人間にも電源スイッチがついていて、パチッと切れば完全休養できたらいいと思ってしまいます。さて、ここで問題です。シャープさんはファミコンのときのようにスーパーファミコンの互換機を発売するでしょうか。答えは編集部の人が聞いてください。
岡部 康(25)福井県
そう、スイッチはほしい。しかし、誰もスイッチを入れてくれなかったときのことを考えると恐い。タイマーをつければいいのかな? それと、シャープさんはスーパーファミコンつきのテレビを発売するようですね。

◆この間、おまけディスクの中から「ナイトアームズのエンディングテーマ」を引っ張り出して聴いてみようかと思って、いろいろいじっていたら、なんとシステムディスクが壊れてしまった。やっぱり慣れないことはするもんじゃない(実は最近ようやくX-BASICを起動させました)。P.S. 6月号の悪魔城伝説のコンフィギュレーションファイルの使い方がわからない。ド素人なもので……情けない。
引間 裕樹(20)大分県
悪魔城伝説のコンフィギュレーションファイルはリスト4をエディタなどで打ち込み「AKUMA.CNF」というファイル名でセーブします。そのあとは記事中の例のようにすれば大丈夫です。



◀佐々木 京都府 憧れのあるデフォルメされたバイクの全体像も見たかったなあ。冬は寒いけど、やっぱりバイクはやめられないし。


◀木下 佐賀県 とってもきれいだけど、クールビューティーです

◆先日、某パソコンショップへひやかしに行くとX68000SUPER-HDの本体が透明なヤツを見つけました。昔、ファミコンのソフトで「スケルトンカセット」なるものがありましたが、ちょうどそのようなものです。「なるほど、本体の中はこうなっているのか」などと感心してしまいました。あれは市販されているのでしょうか。
菅原 伸弘(15)北海道
市販はされてないでしょうね。どうせなら普通のやつよりほしいな

◆つい最近、うちの学校で文化祭をやった。それに出展するためにZ80で制御するF1のプログラムや、対戦テトリスX68000版を作ったりした。だが、しょせんプログラマは裏方なもので、F1のボディや対戦テトリスのゲーム自体にはよろこんでくれるが、プログラムがすごいと誉めてくれる人はいなかった……。プログラムなんてあってもないに等しいからね。しょせん……こんなものよ。
村越 美広(1)静岡県
でも、いきなり知らない人がやってきてプログラムを読みだし「うーん、これはすごい」とか言われても結構恐いものがありますよね

◆東京、自宅で時間のない私は、いまごろ「闇黒の開拓」をやって、最後のほうで行き詰まって悩んでいる。みんな、そんなの忘れてしまって助けてくれないだろうなあ……。と、古すぎる私。
前田 早苗(28)岐阜県
子供が育つまで待って、その子にゲームを解くのを手伝ってもらうというのはどうでしょうか。

◆我が関東大学工学部に今年、UNIXマシン(端末)が入り、後期から演習が始まった。Human68kのCOMMAND.Xしか使ったことのない(MS-DOSはちょっとだけ)僕にとって最初はわけがわからなかった。しかし、最近UNIXのすばらしさがわかり、演習が楽しいんです。まだ、機能のほんのちょっとがわかっただけですが、いつかは完全にマスターしてみせるぜ。
天野 貴明(18)埼玉県
いつか、UNIXを完全にマスターして、家に1台UNIXマシンを買ってください。

◆質問したいのですが、X68000SUPER-HDにはビジュアルシェルが付属されていませんでした。手に入れる方法を教えてください。
鈴原 海義(23)神奈川県
ビジュアルシェルを手に入れる方法ですか? それはですね、今月号を買ってよかったでしょう。


◀舞 山口県 みんなよろしくね 15歳の女の子からの元気イラスト とってもかわいくてとっても気持ちのいいかな



◀北原 岡山県 この人は毎月のように送ってくれますが、別に女の子からといってひいきしているわけではありません(私は少女はしていないけど)

◆11月号は職員旅行出発の日に届いたので旅行へ持っていく。今回の旅行の狙いはできたての高級旅館を視察することで、一泊2万数千円だそうだ。露天の東京館の別館で顧問という。ロビーに着くと琴の生演奏をやっていた。全員に昆布茶が出た。木は全部無ブシの檜だそうで、琴の後ろにあるビョウブは数千万円だそうな。この宿は○百万円で、etc……。室は50畳ほどありまして――しかし、酒をいつものように飲みすぎる者には全部関係ありません。5千円の宿でも同じです。
平山 謙司(48)静岡県
しかし、そんなに高価な品物が周りにあると思うと、落ち着いて飲んでいられなかったのではないでしょうか。

◆使用済みの熱転写インクリボンを巻き戻し、再びプリンタにセットしてベタ打ちする。すると、かつて印字した文章が白ヌキ文字として蘇る。ディスクを処分したから、もう安心と思っているアナタ、秘密文書を誰かに読まれているかもしれませんよ。
神辺
しまった。アレを読まれると非常にマズイじゃないか

◆私は岡市立中学校に通っている17才なのですが、その日は全清掃の柔道があるので、朝から牛乳をたらふく飲んでいきました。この日の柔道ときたら……もうおわかりでしょう。みなさん牛乳は完全食品といわれますが、おなかに消化酵素がないため飲みすぎには十分注意しましょう。
戸谷 滋夫(17)静岡県
柔道があるので牛乳を飲んでいったというのもイマイチ意味がよくわかりませんが、骨折しないようにかな?

◆郵便配達の方に表札を出してくださいといわれた。しかし、立派な表札を作ろうとすると結構お金がかかるわけで、こうなったらワープロで打ってビニールでもかぶせて玄関に貼ろう(情けないです)と早速ピコピコ。さあて印刷だ。ガチャガチャ。なんだこりゃあ。字の真ん中あたりに白く空白の部分が。説明書には印字ヘッドの周囲か、ヘッド周囲回路の故障と書いてある。トホホ。
鈴木 俊明(27)千葉県
表札を買うか、プリンタを直すか、それが問題だ。

◆汚い字ですみません。ちなみに僕は小学校2年生から書道をやっています。でも、それが賞状に生かされません。どうしたらよいのでしょうか。
土井 雅大(1)宮崎県
書道をやめてみる、とか。

◆あごが痛い。歯医者さんへ行ったら、あごの骨がずれて関節炎を起こしているといわれた。せっかくの秋なのに固いものが食えない。ところで、編集室で通産省の情報処理技術者試験に合格している人はどのくらいいるんですか。
本岡 晃(2)愛知県
情報処理試験ねえ、受けろといわれれば受けないこともないですが。ははは。

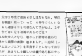
◆Oh!Xのページの下にメッセージがありますが、どうせなら上にも横にも書いたらどうでしょうか。ゲームのちょっとしたテクニックや、役に立つけれど特集にすることもできない話を数ページにわたって書いたりしたらにぎやかになっていいと思うのですが。
宮浦 信明(17)香川県
そんな字だらけで、耳なし芳一みたいな本はやだ。

◆えっ? またディスクがつくんですか。それは楽しみだなあ。なんてったって、私の場合Oh!Xを読み始めたのはあのディスクつきからですもんね。それに、VS.Xだけではなく COMMAND.Xから立ち上げるようになり、Human68kが理解できるようになったのも、あのディスクのおかげです(そういう人って多いと思う)。編集部の人はどう思っておられるのか知りませんが「付録ディスク」はユーザー(Oh!X読者)を成長させます。
下川 毅(28)和歌山県
今回の付録ディスクは誰でも簡単に使えるようにVS.Xが入っているんですが、いろいろと応用的な使い方をしていただければまた「成長」の助けになるかもしれませんね。

◆11月号の巻頭の記事は私の言いたかったことをほとんどすべて代弁してくれたみたいで読んでいて非常に楽しかった。特に、"実際にやりすぎている"というのは実感に共感できるところだ。なんかいまのパソコン上で動いているフライトシミュレータというのは"現実"と"リア







ぼくらの掲示板

仲間

売ります

買います

バックナンバー

ORIVE ON

このコーナーでは、本誌年間モニタの方々のご意見を紹介しています。今回は、こちらの都合でモニタレポートの用紙の発送が遅れてしまい、モニタの方々には大変ご迷惑をかけました。お詫びいたします。今月は11月号の記事に関するレポートです。

■特集「理科系のGAME REVIEW」について。「あなたがゲームを作れない理由」というのは、いままでのゲーム特集記事にないテーマであるが、非常によいと思う。ゲーム特集というのは、悪くいえば技術とは無縁でそのゲームが下火になったらなんの役にも立たない記事になりがちであった。しかし、この記事のお陰で今回の特集が単なる評論に終わらず技術的な分野への橋渡しとなりえたからである。私もあまり人のことは言えないのだが、特に最近はシステムの巨大化、ブラックボックス化、さらに恵まれた市販ソフト環境のために、ユーザーの創作意欲が失われ、あまりプログラミングをしなくなってきているので、こういった記事は積極的に載せてほしい。
泉 岡雄(20) X1turbo model 30 PC-E500 東京都

■ゲーム特集に関して取り上げてほしいテーマ。それは「音楽曲」ですね。よくありませんか? 欧米の秀作が移植されたとたんに「クソゲー」に成り下がってしまったものが。それは「曲」の移植に失敗したからだと思います。下手にグラフィックがよくなってキャラクターデザインやグラフィックを一新してしまい、日本風のグラフィックに西洋風のゲームシステムというミスマッチを生んでしまったゲーム。ちょっと考えるだけでもいろいろ出てくる。

欧米のゲームと日本のゲームを比較してなにが違うかを考えると、やっぱりゲーム自身の「雰囲気」(あやしい雰囲気のもあるけど……)が違うと思う。音楽、グラフィック、システム全部がひとつの雰囲気にうまくまとまっていて、しかもそれが自分の好みにあっていると、やっぱりのめりこんでしまうものでしょう。音楽曲というのはやっぱり大切だと思います。
[illegible] 郷(17) X68000PRO、MSX2 埼玉県

■「文科系のGAME REVIEW」というのもなんか変ですけど、うん、理科系だからこそあのような味のあるレビューが書けるのではないでしょうか。[illegible]文系だの理系だのというのはうんざりですけど、私は味のある文の書ける理系人になりたいと思っています。知人にもちゃくちゃ数学ができるのに、「私、国語なんか全部わからんよ」という人もいますが……。
岡井 [illegible](16) X68000PRO 愛知県

■私が思うに、文科系は万能選手である人(というよりなんにでも関係のある人)を多いが理科系には案外少ないのではないでしょうか。理科系は特殊な分野です。理由がそこにあるからかどうかまで言及できませんが、理科系には「数学」と「理科」だけしかできない人が多くいて、文科系には、得意ではないけども、という条件つきで「数学」や「理科」もできる人が結構いるということを完全に否定できるものではないと思います。

ゲーム誌で特集を組める雑誌もそうそうないでしょうから、こういったOh!Xらしい企画には好感を持ちます。ただ、私の場合、ゲームについての事前知識があまりに少ないものですから、少々難しすぎたきらいもありました。ゲームという、とかく軽く見られがちなものに対して深く考察を入れることは大変興味のあることですし、理科系らしい理屈っぽさも一応の成功だといえると思います。

ゲーム論理といったものから、ごく限られた一点にのみテーマをしぼったものまで、偏りもなく、バランスよくまとまっていました。また、「オレにはどうせわからせんねん」という読者にも助けられて気楽に楽しく読ませてもらいました。
浅野 [illegible](19) X68000PRO X1turboIII、X1F model 20 MZ-80C、FM 77L2、MSJr PC-500 PC-1245 大阪府

■泉さんの「将棋はパソコンである」は清水さんや祝さんや[illegible]さんのような記事ですね(なにか似ている)。それはいいとして、この記事のよかったところは、マシン自身になりきった点であり、これはシャープユーザーに多いパターンであるからだ。記事を読んでいてマシン自身になれた気分になる。かなりすごい(?)記事になっている。もう一度読み直してみようっと。
柚部 裕振(18) MZ-1500 北海道

●「THE SOFTOUCH SPECIAL」について。前半のティグナスの冒険までは特にいつもと変わってないと思う。後半の機種以降の記事は買う側にとってみると、とても便利。実際にショップでいつも見比べたりする必要がないし、どこが違うのかをすぐに把握できる。半年に一度くらいずつ、ゲームのジャンルごとの比較をやって欲しい。
[illegible] 剛[illegible](18) X1turbo model 10、X1turboZ II MSX2、JR-100 北海道

●第100回を迎えたS-OSについて。昔は「C言語は8ビットに向いていない」といわれ(いって)SLANGが生まれた。C言語もできそうな雰囲気であるが、やはりS-OSにはSLANGであると思う。僕もS-OSでシステムは組んでいるが、残念なことに毎月すべて打ち込むといったところまではいけない。忙しすぎるのだ。しかし、S-OSのシステムはいまだにすたれてはいないと思う。このS-OSの精神はいまのコンピュータ社会にはないものがある。この開発精神こそ、本誌はすべての"コンピュータで遊ぶ人"に持っていてもらいたいものである。掲載についてはこのような月1本ペースをくずさないでもらいたい。
長谷川 敦士(17) MZ-2500、MSX2 山形県

ごめんなさいのコーナー

12月号 THE SOFTOUCH SPECIAL
P.46 イメージファイトの記事で価格が間違っていました。2,800円ではなくて、8,800円(税別)です。関係者の皆様には大変ご迷惑をかけました。

12月号 INTEGRAL X1
記事中ではリスト1が「ME.BAT」、リスト2が「MENU.X1」になっていますが、実際のリストでは逆になっています。リスト中のファイル名を参照してください。申し訳ありませんでした。

バグに関するお問い合わせは
☎03(5488)1311(直通)
月〜金曜日16:00〜18:00

お問い合わせは原則として、本誌のバグ情報のみに限らせていただきます。入力法、操作法などはマニュアルをよくお読みください。また、よくアドベンチャーゲームの解答を求めるお電話をいただきますが、本誌ではいっさいお答えできません。ご了承ください。

謹賀新年PRO-68K 次はいったい?

▼2度目の付録ディスクでしたが、いかがでしたでしょうか。いわゆる特典版ではなく、自前で制作しているオリジナルフロッピーですから、どうしても原価が高くなってしまいます。どうせ高くなるのならと、そのぶん内容は盛り沢山。97ページのファイル名一覧を見ていただければその凄さがおわかりでしょう。収録したものは皆さんに"見てもらいたいもの、使ってもらいたいもの"ばかりです。まだまだ入れたいと願いながらも、容量の都合で次回まわしになったものもあります。というわけで次回のディスクは4月号を予定しています(どうしても780円は高いという方には、やはり定期購読をお勧めします)。

さて、ディスクにはある程度の大きな規模のプログラム(ゲームやツールなど)が収録できますので、皆さんからの作品をお待ちしています。もちろん採用されたものには原稿料をお支払いいたします。ぜひともオリジナル作品を投稿して、X68000の元手ぐらいは回収しましょう。

▼待望のSX-WINDOWの資料が配布できることになりました。付録ディスクに収録された膨大な資料はソフトハウスなどの技術者向けに書かれたものを編集部でまとめたものです。そこで特集もSX-WINDOWに関する解説となりました。資料と共にご活用ください。また、ウィンドウ上でのアクセサリは今後とも付録ディスクなどで配布できるよう考えていますとりあえず、アナログ時計やカレンダーなどアクセサリを制作した方はぜひとも投稿してください。

▼村田敏幸さんの「X68000マシン語プログラミング」が単行本化され、ソフトバンクの書籍編集部から発売となりました。連載の入門部に加筆訂正を加えたものです。詳しくは183ページのお知らせをご覧ください。

▼今月もページの都合でいくつかの連載がお休みになってしまいました。「マシン語カクテルin Z80's Bar」「X68000マシン語プログラミング」はお休みです。

投稿応募要領

- ●原稿には、住所、氏名(年齢)、職業、連絡先電話番号、機種、使用言語、必要な周辺機器(マイコン歴を明記してください。
- ●プログラムを投稿される方は、詳しい内容の説明、利用法、できればフローチャート、変数表、メモリマップ(マシン語の場合)に、参考文献を明記し、プログラムをセーブしたテープ(ディスケット)を添えてお送りください。また、掲載にあたっては、編集上の都合により加筆修正させていただくことがありますのでご了承ください。
- ●ハードの制作などを投稿される方は、詳しい内容の説明のほかに回路図、部品表、できれば実体配線図も添えてください。編集部で検討のうえ、製作したハードが必要な場合はご連絡いたします。
- ●投稿者のモラルとして、他誌との二重投稿、他機種用プログラムを単に移植したものは固くお断りいたします。

あて先
〒108 東京都港区高輪2-19-13 NS高輪ビル
ソフトバンク出版部
Oh!X「㋢㋙㋮㋫」係

S H I F T ・ B R E A K

▶田舎嫌都営御隅駅前(ややこしい)で、[illegible]警備のおまわりさんに声をかけられ、ついつい話がその道に……、そのおまわりさん、かつてはバンドをやっていて、スティーブ・コイが心の師だったとか「今は弾いてないんですか?」と聞いたら「吹いてるよ、"君が代"を」だって(笑)。先月とはうってかわって警察が好きになってしまった。(須)

▶江戸っ子シリーズ第2弾。あろうことか私は「熱い風呂とぬるい風呂」とでは、ぬるいほうが好きなのだ。通湯などという熱湯にはとてもじゃないが入れない。長年、私の「風呂」は「江戸っ子でない証拠」とされてきたのだが、ひとつ攻撃として「べらんめぃ!」の「べ」で舌をまわせることを提唱しておこう。(亀)

▶早いもので、この編集室に私が来るようになってもう3度目の冬がやってきた。時の流れは人を変えていく。それはかつてオムライスの(で)と呼ばれた私にあってもまたそうであった。最近は大阪、四谷の近く満足感、熱かんきゅーっと一杯、ああ、[illegible]うどんのうまい季節。
(単に寒いからじゃないのか? それは。(で))

▶雪が降ましたといっても私が降らしたのではありません。雪が降らないのもきっとその世いではないでしょう。スキーに行く予定がある私はとっても悲しい思いをしています。雪をブーンもウェアも必要がないでしょう。酒とつまみとゲームさえあれば、ほかになにもいらないスキーツアーになりそうです。ところで、まかったね〇〇君。(S.K.)

▶僕はその昔いが大嫌いだ。バイクの機嫌は悪くなるし、凍っていてを寒い、楽しみなイベントとてない。え? クリスマス? な一しそれ(空しい)。さて、これを書いている今は11月下旬。が、例年に、なく暖かく、スキー好きの人々が嘆きめているのを聞くに、スキーをしたことのない僕は、密かに、ことを単純に喜んでいるのだ。ああ不健康。(AT)

▶あのですね、僕は不信感を募らせつつあるのですよ、シャープさん。まずノーザーズカップからおみこしに至る、必要以上に青少年を煽った広告にはまりましょう。不快ですけど、SX-WINDOWを許します。真摯に作っているのは伝わるから。でもXDだけはなんとかしてよ。自社製品の規約によその使うのって悲しいでしょ?(Mu)

▶引っ越しをした。外国人不可や水商売不可、子供不可や商談不可というのは聞いたことがあった。しかし、事態はここまできている「[illegible]」「一部上場企業社員のみ」ときた。ひでえ話だが、東京は、こういう状態なのだ。私のような自由業勤め先ナンは信用するに値しないらしい。東京は隔離されたりをしているだけなのね。(K)

▶ファミコンは昔から持っていたが、ディスクシステム専用の「ボディコンクエスト」をプレイするため、ツインファミコンを買った(今さら)。しかし、ゲーム自体の操作性の悪さのため、すぐ放り出して知人に譲ってしまった。すると知人は1日で解いて詳しい地図まで描いてくれた。RPG、は忍耐力が必要だったことを再認識した今日この頃。(KO)

▶この間、AMIGAのソフトでも買おうかと思い、輸入所のそのテの店に[illegible]を出してみた。ちょうど入荷日だったらしく、ソフトがドカッと入ってきたんだけど、そこにはなんと、あの「シムアース」が山のように。残念ながらMacintoshは持っていないので関係ない。それよりアレはまだかなあ。ずいぶん待っているんだけどねえ、モナミューさん。(A)

▶疲れたなあ、なんか眠いや。あ、身体が軽くて気持ちがいい、そうか、あたし死ぬんだね。思えばいっぱいいろんなことがあったなぁ、みんな、さよなら、おやすみ……、……ん? あれ? 生きてら。なぁんだ、死んだんじゃないのか。神サマの思し召しかな、しゃ、これからは世のため人のためにこの身を捧げることにすっか。(生まれ変わったE.O)

▶仕事のあいまに編集長がゲームをやっている。「この画面を見てください! 世界に誇れるゲームに見えますか?」見えるぞ「不謹さと実験は一切しておりません」本当だなぁ? マシン語に出慢したAMIGAではIndianapolis 500が人気。速度と画面が重いと思っていたら、サウンドやゲーム性はもっと重いことがわかった。(U)

▶だから先月はあらかじめ言い訳を書いておいたのですが……、ところで、X68000の最大の魅力ってなんでしょう。私はそれぞれのユーザーがパソコンの可能性を模索できる自由さにあると思うのです。でも、メーカーさんにはもっと強烈なイメージを与えてほしいですね。ユーザーをあてにすることを表明するようなプロパガンダには寂しくなります。(T)

microOdyssey

今回のディスクは「承」のディスク！

前回の創刊8周年記念PRO-68Kがどうしても持っていてほしいものを集めたディスク、いまなお大きな意味を持っていることはいうまでもない。Oh!Xにとってある意味で「起」のディスクといえる。今回はそれを継ぐものとなる。

いや、むしろ方向性は異なる。前回のディスクを作成したときにも書いたが、そのとき選ぶこともできた「別のアプローチ」というもの、それに近い。単にプログラムの集合体ではなく、ひとつのメディアであり「場所」であるもの、それが今回のディスクだ。

個人的にSX-WINDOWはシステムとしての意味以上に「場所」としての意味が大きいと思っている。システムのパフォーマンスで計るならウィンドウ環境が他のウィンドウ環境に勝てるはずはない。しかし「場所」であることがもたらす力はもっとも強い力といえる。

昔、S-OSをシステムのパフォーマンスでとらえた人は離れていき、場所としてとらえた人たちは、これまでに見られるだけのものを築きあげた。パフォーマンスという面では、S-OSはこれ以上ないというくらい不利な位置からスタートしたシステムだ。相対的にいままでは他のS-OS環境よりも総合的なパフォーマンスが向上しているといっていいだろう（ただ、X68000のS-OSエミュレータ上でのWINERよりも表示の遅いSX-WINDOWのノートはなんとかしてほしい……）。

問題は「場所」をどのように設定するかだ。別にOh!Xでは、通信をやっていないユーザーのためにディスクをつけているわけではない。パソコン通信のネットで出回っている公開ソフトを入手したければ、それはパソコン通信を始めるべきだ。

一般読者に掲載されたプログラムはあまり重視しないという方針に変わりはない。入力しにくく、かつ、誰にでも持っていてもらいたい「評判」なもの、「紙面」には掲載できない大きさの投稿作品などを除けば「Something New」を追求する、それが基本方針といえる。

さて「承」があれば「転」があるだろうと予測する人は多い。「結」はあまりないだろうから「転」が完成形を兼ねるとしよう。今回が前回に対して「承」であり、前回に「起」であるなら「結」は「起承結」すべてを兼ねることになるだろう。これでサイクルは完結する。問題はこれを満たす「力」。

スタッフは待機している。

「場所」は作られた。あとは「力」が集まるのを待つしかない。

紙面とディスク容量に余裕があればもっと扱いやすくできたと思う。2HDの1.2Mフォーマットならもう一枚ディスクが増えただろう。

前回のマスターディスク作成はRAMディスクとハードディスク（これもかなり贅沢な環境だったのだが）で行われたが、今回の付録ディスク作成はずいぶん強化された環境で行われた。

さっそく光磁気ディスクには大活躍してもらった。HAL研のハンディスキャナと加工ノールには本当にお世話になった。120Mバイト分に増設されたハードディスクは強力だった。

しかし、もっとも活躍したのはシステムが更新されるたびに初期化されていった数十枚のフロッピーディスクだったかもしれない。(U)

1991年2月号 1月18日(金)発売

特集1 "実験的"グラフィック入門

追加解説 Z's-EX/HASH.X

特集2 SX-WINDOWへの第一歩

1990年度GAME OF THE YEARノミネート発表

Oh!X LIVE in '91

Z-BASIC用 サザエさん劇中曲

X68000用 Misty Blueオープニング

バックナンバー常備店

東京	神保町	三省堂神田本店5F	03(233)33[illegible]
	〃	書泉ブックマート6	03(294)[illegible]
	〃	書泉グランデ5F	03(295)0011
	秋葉原	T-ZONE 7Fブックゾーン	03(257)[illegible]
	八重洲	八重洲ブックセンター3F	03(281)[illegible]
	新宿	紀伊國屋書店本店	03(354)[illegible]
	[illegible]	[illegible]	03(20[illegible])[illegible]
	渋谷	大盛堂書店	[illegible](463)0511
	池袋	リブロ池袋店	03([illegible])[illegible]
	〃	[illegible] コンピュータ フォーラム	03([illegible])[illegible]
神奈川	横浜	有隣堂[illegible]	[illegible]
	〃	有隣堂[illegible]	[illegible]
	[illegible]	有隣堂[illegible]	[illegible]
神奈川	厚木	有隣堂厚木店	0462(23)4[illegible]
	平塚	文教堂[illegible]	0463(54)2880
千葉	[illegible]	[illegible]	[illegible]
	船橋	リブロ船橋店	0474(25)[illegible]
	〃	[illegible]	[illegible]
	千葉	多田屋千葉セントラルプラザ店	[illegible]
埼玉	[illegible]	[illegible]	0492(25)[illegible]
	[illegible]	[illegible]	[illegible]
茨城	水戸	[illegible]	0292([illegible])
大阪	[illegible]	[illegible]	[illegible]
	[illegible]	[illegible]	[illegible]
京都	中京区	オーム社書店	075(22[illegible])
愛知	[illegible]	[illegible]	052(562)0077
	〃	パソコン[illegible]	052(251)[illegible]
	[illegible]	[illegible]	0566([illegible])1134
山梨	[illegible]	[illegible]	0265(24)[illegible]
北海道	[illegible]	[illegible]	0143(44)[illegible]

定期購読のお知らせ

Oh!Xの定期購読をご希望の方は誌上込みの振替用紙の「申込欄」欄にある「新規」「継続」のいずれかに○をつけ、必要事項を明記のうえ、郵便局で購読料をお振り込みください。その際渡される半券は領収書になっていますので、大切に保管してください。なお、すでに定期購読をご利用の方には期限終了の少し前にご連絡いたします。継続希望の方は上記と同じ要領でお申し込みください。

海外送付ご希望の方へ

本誌の海外発送代理店、日本IPS(株)にお申し込みください。なお、購読料金は郵送方法、地域によって異なりますので、下記へ必ずお問い合わせください。

日本IPS株式会社

〒[illegible] 東京都千代田区[illegible]3-11-6

☎03(238)[illegible]

Oh!X 1月号

■1991年1月1日発行 定価780円(本体757円)

■発行人 孫 正義

■編集人 橋本[illegible]

■発売元 ソフトバンク株式会社

■出版事業部 〒105 東京都港区高輪2-19-13 NS高輪ビル

Oh!X編集部 ☎03(5488)1389

出版営業部 ☎03(5488)1360 FAX 03(5488)1364

広告センター ☎03(297)0181

■印刷 凸版印刷株式会社



満開の電子ちゃん

作・いわいいっぺい
え・岡村 祭



おかげさまで満10歳

X BASIC HOUSE PRO SHOP

豪華絢爛10周年特別販売

新登場!

SCSIインターフェース標準装備

X68000 SUPER
(HDD無し) CZ 604C
標準価格¥348,000

さらにお求め安くなった

CZ-606D-TN /BK/GY
ドットピッチ0.31mm
CZ-603D後継機
標準価格¥79,800
続々入荷!!

memory 6M set

余裕の6Mバイトメモリー
EXPERTII



CZ-603C
CZ-613D
KGB-X68PRK-04
合計金額¥595,000
10周年特価¥468,000

PROII



CZ-653C
CZ-603D
PIO-6BE1A
KGB-X68PRK-04
合計金額¥516,800
10周年特価¥405,000

音遊MIDI!

ローコストセット
CM 32L SX-68M Mu 1
BH特価¥89,800

本格音源セット
CM 64 SX 68M Mu 1
BH特価¥139,800

新一体型スピーカーセット
CM 32L CS-10 SX-68M Mu 1
BH特価¥98,800

最高級

10周年限定大特価
特別セットのため台数限定です



CZ-613C TN
CZ-603D TN
GT-4000
KGB-X68PRK-14
合計金額¥991,000

super price ¥698,000

大容量HDD

SASI type
LOGTEC SHD-040(40M) ~~¥ 98,000~~

SCSI type
NEC PC HD040L (40M) ¥ 94,800
ITEC FC H100 (100M) ~~¥158,000~~

注目! その3

136桁インクジェットプリンタ
特価¥69,800

80桁ドットインパクトプリンタ
特価¥39,800

※台数に限りがあります
※ケーブルは別途お買い求め下さい

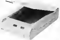
電子手帳

PA 9500	定価¥48,000	BH特価販売中!!
PA 8600	定価¥28,000	BH特価¥24,800
CE-200L	定価¥2,500	BH特価¥ 2,200
CE 300L	定価¥2,800	BH特価¥ 2,500
CYBER NOTE PRO-68K		定価~~¥19,800~~
Stationary PRO-68K		定価~~¥14,800~~

注目!

台数限定早いもの勝ち!

EPSON GT-4000



super price ~~¥118,000~~

注目! その2

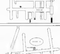
10周年記念台数限定スキャナーセット

SUPER HD SET
CZ-613C TN
CZ 613D TN
GT-4000
特価 ¥610,000

EXPERTII SET
CZ 603C
CZ 613D
GT-4000
特価 ¥488,000

PROII SET
CZ-653C
CZ-603D
GT-4000
特価 ¥398,000



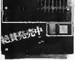


X68000のことなら

PRO SHOP/STAFFにおまかせ下さい
価格の事から技術的な事まで何でもご相談ください
技術的なお問い合わせは個根まで
通信販売担当は豊嶺　神山、片桐です。

2枚のボードが1枚になった

KGB-X68PRK



広大なメモリ空間を実現する最大4Mバイトの
高速増設メモリ

高速演算を約束してくれる
数値演算プロセッサ

- ■メモリアクセスノーウェイトによる超速アクセス
- ■CZ-6BE2 CZ-6BE4 CZ-6BP1との混在が可能
- ●複数枚のKGB X68PRKの実装が可能
- ●ジャンパの変更によりお客様のアドレス空間にメモリの配置が可能
- ●ジャンパの変更により数値演算プロセッサの1枚目 2枚目 未使用の選択が可能
- ●1M 2M 3Mメモリモデルは購入後もメモリ増設が可能
- ●PRK 10, 11 12 13, 14にはデバイスドライバ(FLOAT3 X)が付属

●CZ 600C, 601C 611C 652C 653C 662C, 653Cで御使用の際にはあらかじめ専用の1Mメモリ(CZ-6BE1 A 等)でメインメモリを2Mバイトにしておく必要があります

製品価格一覧

KGB-X68PRK-00	¥34,000
KGB X68PRK-01	¥58,000
KGB X68PRK-02	¥74,000
KGB-X68PRK-03	¥98,000
KGB-X68PRK-04	¥122,000
KGB-X68PRK-11	¥95,000
KGB-X68PRK 12	¥112,000
KGB-X68PRK-13	¥138,000
KGB-X68PRK-14	¥160,000

購入後の増設費用

メモリ	
1Mバイト	¥24,000
2Mバイト	¥51,000
3Mバイト	¥76,000
数値演算プロセッサ MC68881RC16	¥38,000

PRK Q&A

Q 購入後のメモリ増設はどうやるのでしょう?
A ご購入後のPRKに対するメモリの増設は半田付け等の技術を要するため原則として当社に送り返していただき増設いたします。自分でメモリ増設をする場合は通信販売のみですが必要な部品の販売も致します。御希望の方はお問い合わせ下さい。

Q 数値演算プロセッサにMC68882を使用することは可能ですか?
A MC68882では動作しないソフトが存在するため使用できません。

Q 「数値演算プロセッサのみ」や「プロセッサ無しメモリ無し」のPRKがほしいのですが?
A PRK 10 PRK-00の型番で商品化しております。

※最近PRKをスロットに挿入したが動作しないと言う御質問を良く受けますが、ほとんどの場合は差し込み不足が原因です。X68000のスロットは大変堅く隙間が閉まる状態でも差し込み不十分の場合があります。御注意ください。

充実のBASIC HOUSEソフトウェア&ハードウェア

高速12BIT、16CH A Dコンバータボード(KGB AD12) X1	¥118,000
フォトアイソレーション16BITデジタル入出力ボード(KGB PIO) X1	¥ 42,000
アイソレーション16BITデジタル入出力ボード(KGB X68PIO) X68000	¥ 68,000
ハンディプリンタ&インターフェース(HANDYPRINTjack) X68000	¥ 24,800
高速12BIT, 4CH D Aコンバータボード(KGB DA4) X1	¥ 98,000
汎用ローコストA D&PIOボード(KGB-X1S) X1	¥ 19,800
高速12BIT, 16CH A Dコンバータ(KGB X68ADC) X68000	¥128,000
64180CPUボードMach 180(KGB CPXB) X68000	¥ 98,000
ローコストMIDIインターフェース(MELODY BOX) X68000	¥ 16,800

BASIC拡張関数パッケージ(B6 6301)	¥9,800	C言語ライブラリ(B6 6305)	¥6,800
ディスクキャッシャー(B6 6304)	¥6,800	Toys & Tools (B6 6307)	¥6,800
BASIC拡張関数パッケージC言語ライブラリ付(B6 6306)			¥14,800
アイコンエディタ(B6-6303)	¥4,800	CP M68Kエミュレータ(B6-6302)	¥19,800

おしらせ

DISK CACHER Version UP

価値に御愛用いただいているディスクキャッシャーが高速化(従来比平均3倍)を行ないVer UPいたしました。今回のVer UPはハードディスクキャッシャーのみでHD-DISKCACHE Ver 2.0未満のキャッシャーを御持ちの方がVer UPの対象となります。
御希望の方は旧バージョンのディスクのラベルと代金¥1,500(送料、税込み)を同封して現金書留で御申し込み下さい。

ビデオボードを外付けに!!
ビデオボードケース(KGB-BVBX)
大好評発売中

SHARPより発売されているCZ 6BVIを外付けにするケースです。このケースの使用によりあなたのX68000のスロットが開放されます。

Human68k下のソフトのCRT出力を強制的に15kHZ出力にする(768×512モード除く)おまけユー

全国どこでも発送可　各種クレジットOK　送料全国一律¥1,000　宅配便にて即日配達

株式会社計測技研　本社営業部　マイコンショップ／通販部　宇都宮市竹林町503-1　TEL.0286 22 9811　FAX0286 25 3970
大田原営業所／マイコンショップ　大田原市美原1-13-4　TEL.0287 23 5352　FAX0286 23 5364
マイコンショップ BASIC HOUSE　お申し込み・お問い合せは ☎0286-22-9811(代)

AIBIT
アイビット電子株式会社

SHARP
パソコン本体から周辺機器まで品数取り揃え
大特価セール実施中!!

型名	品名	正価	特価
PC-E200AL	ポケコン	20,800	18,000
PC-1600K	ポケコン	60,000	43,000
PC-1300K	ポケコン	38,000	32,000
PC-1360	ポケコン	29,800	19,000
PC-1262	ポケコン	24,800	18,000
PC-1248DB	ポケコン	11,800	9,000
PC-1280	ポケコン	24,800	18,000
CE-T800	ポケコン用I/O・232Cコンバーター	12,000	11,000
CE-203M	ポケコンRAM32K	32,000	7,000
CE-202M	ポケコンRAM16K	35,000	8,000
CE-201M	ポケコンRAM 8K	16,000	3,000
CE-1600M	ポケコンRAM32K	32,000	18,000
CE-1600F	ポケコンフロッピードライブ	39,800	34,800
CE-1600P	ポケコンプリンター	29,800	29,800
CE-1650F	ポケコンDISK	8,800	8,000
CE-161	ポケコンRAM16K	58,000	3,800
CE-1601M	ポケコンRAM64K	45,000	30,000
CE-1600E	ポケコンディスクインターフェイス	19,000	17,800
CE-160	ポケコンレベルコンバーター	39,000	31,300
CE-159	ポケコンRAM 8K	35,000	4,200
CE-140T	ポケコンRS-232Cコンバーター	5,000	1,000
CE-140F	ポケコンフロッピディスク	49,800	44,800
CE-126P	ポケコンプリンター	19,800	17,800
CE-129P	ポケコンプリンター	24,800	21,800
CE-124	ポケコンカセットインターフェース	4,500	4,000
Z VISION	[illegible]	[illegible]	51,000
UX-1	ホームコピーファクス	79,800	63,000
PA-5600	ハイパー電子手帳	[illegible]	38,000
CZ-300F	X1ターボマイクロフロッピー	78,000	8,000
CZ-01FB	300F増設フロッピー	59,800	7,000
CZ-30F	CZ-802C増設フロッピー	59,000	9,000
CZ-501H	X1増設用ハードディスクユニット	258,000	86,000
CZ-503F	CZ-増設ドライブ	41,800	38,000
CZ-520F	フロッピードライブ	118,000	70,000
CZ-8BG1	GPIBボード	58,800	47,800
CZ-8SP1	数値演算ボード	79,800	63,800
CZ-8BC1	FAXボード	79,800	60,000
CZ-8BU1	ユニバーサルI/Oボード	80,800	33,800
CZ-8BM1	MIDIボード	29,800	23,800
CZ-5BE1A	1M増設RAMボード	39,800	19,500
CZ-6BE1B	1M増設RAMボード	29,800	18,800
CZ-6BE1	1M増設RAMボード	35,000	28,000
CZ-6BE2	2M増設RAMボード	79,800	特価
CZ-6BE4	4M増設RAMボード	139,800	特価
CZ-6BN1	スキャナーボード	29,800	26,500
CZ-6BF1	RS-232C増設ボード	45,800	42,300
CZ-6BD1	システムラック	44,800	58,000
CZ-8TU	RGBシステムチューナー	33,100	26,500
CZ-822C	X1G MODEL30	118,000	30,000
CZ-820C	X1G MODEL10	89,800	18,800
CZ-880C	X1TURBO	——	特価
CZ-8GR2	グラフィックボードX1	14,800	3,000
CZ-8BF1	FDインターフェイス	14,800	11,800
CZ-8BK2	X1漢字ROM	11,800	10,800
CZ-8BM2	232Cマウスボード	11,000	18,800
CZ-8BE2	320K外部メモリー	29,800	26,300
CZ-8BR1	立体映像セット	34,000	33,800
CZ-8BV2	カラーイメージボード	39,800	32,900
CZ-1BD1	FDインターフェイス	14,800	8,000
CZ-1BT2	モデムユニット	88,800	38,200
CZ-1EB3	増設I/O BOX	33,800	26,800

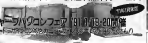
型名	品名	正価	特価
CZ-8LM1	232Cケーブル	7,200	6,000
CZ-8LM2	232Cクロスケーブル	7,200	6,000
CZ-8NJ1	ジョイカード	1,780	1,300
CZ-8NT1	トラックボール	13,800	11,500
CZ-8PK10	24ドット136桁漢字プリンター	[illegible]	[illegible]
CZ-8PK7	24ドット80桁漢字プリンター	122,000	58,800
CZ-8PC4GY	24ドット80桁カラー漢字プリンター	[illegible]	51,000
AN-S100	アンプ付スピーカー	26,800	24,500
HXD040	アイテム40Mハードディスク(ITM)	118,000	96,000
HXD140	40Mハードディスク内蔵用(ITM)	90,000	78,000
CU-14FD	カラーディスプレーブラウン管	74,800	49,800
MZ-1D18	12"モノクロディスプレー	41,800	25,000
MZ-1D17	15"CRT	184,000	60,000
MZ-1E05	MZ-2000 FDインターフェイス	24,800	10,000
MZ-1E08	プリンターI/F	9,800	8,000
MZ-1E11	MZ-6500用 SFD I/F	39,800	28,000
MZ-1E04	MZ-2000プリンターI/F	10,000	8,000
MZ-1E21	MZ-5500 GP I/F	38,000	12,000
MZ-1E19	MZ2000QD用インターフェイス	9,800	3,000
MZ-1E33	MZ6500パラレルI/F	34,800	28,000
MZ-1E45	MZ6500 232C I/F	50,000	15,000
MZ-1E30	MZ2500 パラレルI/F	30,000	27,000
MZ-1E44	MZ-6500 S-RN I/F	50,000	15,000
MZ-1E22	MZ-5500 GP-B I/F	72,000	25,000
MZ-1E20	RS-232Cインターフェイス	17,800	8,800
MZ-1E01	MZ-3500 232Cボード	29,800	13,000
MZ-1E14	MZ-1500 QD用インターフェイス	9,800	3,000
MZ-1M01	MZ-2000/2200 ボード	79,000	9,000
MZ-1M06	演算プロセッサ	[illegible]	30,000
MZ-1M03	MZ-5500 演算	69,000	38,000
MZ-1M12	MZ-2861 演算プロセッサ	80,000	特価
MZ-80P6B	136桁ドットプリンター	——	48,000
MZ-1P06	ドットプリンター	234,000	45,000
MZ-1P26	ドットプリンター漢字	140,000	118,400
MZ-1P16A	24ドットプリンター	246,000	79,800
MZ-1P22	熱転写漢字プリンター	59,800	25,800
MZ-1P28	漢字プリンター136桁	168,000	134,400
MZ-1P30	136桁プリンター	228,000	100,000
MZ-1R01	MZ-2000/2200 IOボード	31,800	10,000
MZ-1R10	MZ-5500 漢字ROM	33,000	8,000
MZ-1R00	MZ-5500 V RAM	35,000	15,000
MZ-1R08	MZ-5500 RAM	45,000	8,000
MZ-1R12	RAM	38,000	8,000
MZ-1R11	MZ-5600 256K RAM	80,000	38,000
MZ-1R38	MZ-2861 1M増設RAM	45,000	15,000
MZ-1R36	MZ-2861 1M増設RAM	55,000	19,000
MZ-1R14	MZ-5500 ROM	40,000	22,000
MZ-1R15	MZ-5500 128K RAM	30,000	8,000
MZ-1R27A	MZ-2500 VRAM	13,000	10,000
MZ-1R26A	MZ-2500 増設RAM	15,800	12,000
MZ-1R21	漢字ROM	38,000	13,000
MZ-1R24	MZ-1500 ROM	22,000	8,000
MZ-1R32	MZ-5500 RAM	80,000	48,000
MZ-1R31	漢字ROM	28,000	28,000
MZ-1R23A	MZ-2500 ROM	13,000	10,000
MZ-1R29	MZ-1P22 RAM	32,000	12,000
MZ-1S13	MZ-1D17 スタンド	12,000	8,000
MZ-1T02	MZ-2000 データーレコーダー	19,800	8,000
MZ-1T03	MZ-5500 データーレコーダー	12,000	8,000
MZ-1U08	MZ-2500 拡張ボード	9,800	7,200
MZ-1V01	パソコンFAX	270,000	98,000

型名	品名	正価	特価
MZ-1X22	モデムユニット	21,800	13,000
MZ-2Z016	MZ-5500 TODAY	25,800	16,000
MZ-2Z036	MZ-5500 附属	——	5,000
MZ-2Z028	MZ-5500 MSDOS GWBASIC	[illegible]	36,000
MZ-2Z023	MZ-5500 GWBASIC	50,000	30,000
MZ-2Z031	MZ-6500 日本語ワープロ	[illegible]	15,000
MZ-2Z025	MZ-6500 TODAY	88,000	20,000
MZ-2Z004	MZ-5500 RAM	68,000	28,000
MZ-2Z005	MZ-5500 RAM	145,000	16,000
MZ-2Z012	MZ-5500 附属	——	5,000
MZ-2Z013	MZ-5500 MSDOS	25,000	20,000
MZ-4Z001	MZ-5600 IBM変換	37,000	8,000
MZ-5621	本体	300,000	50,000
MZ-5611	本体	298,000	36,000
MZ-6Z013	MZ-1500 QDソフト	——	5,000
MZ-1F03	ブランクQD DISK	450	400
MZ-1P10	MZ-1P10用カットシートフィーダー	80,000	36,000
MZ-1P11	MZ-1P10 カットシート	35,000	30,000
MZ-1P21	MZ-1P21 カットシートフィーダー	50,000	37,500
MZ-1P27	MZ-1P27 カットシートフィーダー	50,000	38,000
MZ-1P06	MZ-1P06 トラクターフィード	16,000	7,000
MZ-1P20	MZ-1P20 ホルダー	3,000	2,700
MZ-6Z22	MZ-80 CP/M	18,000	8,000
MZ-6Z26	MZ-80 ストリーマーユーティリティZプロセッサ	80,000	10,000
MZ-80T26A	MZ-80 マシンランゲージ	8,000	5,000
MZ-80TUB	MZ-80 バックアップ	20,000	8,000
MZ-80TU	MZ-80 システムプログラム	20,000	8,000
MZ-80T46A	MZ-80 PASCAL	10,000	5,000
MZ-80T73A	MZ-80 FDOS	20,000	7,000
MZ-80BGK	MZ-80 BGRAM2	35,000	10,000
MZ-8B104	MZ-2000/2200 GPインターフェイス	45,000	16,000
MZ-80BG	MZ-80 BGRAM1	30,000	10,000
MZ-8BC	MZ-2000/2200 GPケーブル	18,000	8,000
AX386-FH8	MZ-8707A	1,280,000	特価
AX386S-F	MZ-8702B	688,000	特価
AX386-F	MZ-8702A	860,000	特価
AX386S-FH4	MZ-8706B	780,000	特価
AX386-FH4	MZ-8706A	1,100,000	特価
AX286L-FH3	MZ-8353A	588,000	特価
AX286L-F	MZ-4052A	421,000	特価
AX286D-F	MZ-8302A	278,000	特価
AX286D-FH4	MZ-8306A	458,000	特価
UE-1U01	AX286L スロットBOX	5,000	4,000
UE-1R02	4M RAMボード	500,000	特価
UE-1R06	ROMボード	32,800	28,800
UE-1R01	2M RAMボード	198,000	特価
UE-1R05	グラフィックボード	98,000	68,000
UE-1R03	1M RAMボード	100,000	特価
UE-1R04	2M RAMボード	180,000	特価
UE-1P03	80桁漢字プリンタ	——	特価
UE-1P04	136桁漢字プリンタ	——	特価
UE-1P05	136桁漢字水平プリンタ	——	特価
UE-1P02	136桁漢字プリンタ	550,000	特価
UE-1P01	136桁漢字プリンタ	280,000	特価
UE-1E04	S-RNインターフェイスカード	79,000	特価
UE-1E02	AX286L ICカード1	45,000	特価
UE-1E03	5"FDインターフェイスカード	28,000	特価
UE-1D03	15インチカラーディスプレイ	123,000	特価
UE-1D02	14インチカラーディスプレイ	158,000	特価

ポケコン関係周辺機器サプライ製品及シャープ関係のソフトウエア全種取扱います。

FM TOWNS/FM NOTE/東芝ダイナブック、周辺機器も取扱っております。

年末謝恩セール実施中!!
セール期間中につき、送料無料サービス



'91年1月末迄

シャープパソコンフェア '91 1/19・20開催
〈カードデザインコンペや、カードデザインスクールなど盛りだくさん!〉

PIXELA
Mac IIフルカラー・イメージクリエータ
〈ピクセカラー735〉
定価 ¥128,000
新発売/入荷



SHARP
光磁気ディスクドライブ
JY-7DG0
新発売/入荷



AIBIT
アイビット電子株式会社

ワープロ,パソコンお買い上げの方はワープロ,パソコン教室が御利用になれます。初・中・上級 無料にて実施中。

SHARP X68000シリーズ対応 ハードディスク
〈ITEM〉
HXD 040 X68000
定価¥ 特価¥95,000
HXD 040 X68000
定価¥ 特価¥102,500
HXD 100 X68000 内蔵用
定価¥ 特価¥79,800

SHARP X68000 特価表示は、オープン記念特別価格にてご提供いたします。
特価表示はTELにてご確認下さい。

CZ-603C(本体)
プラス(ディスプレイ)組合せ

CZ-602DBK	特価
CZ-606D	特価
CZ-611DGY	¥300,000
CZ-613D	特価

CZ-652C(本体)
プラス(ディスプレイ)組合せ

CZ-602DBK	¥275,000
CZ-606D	¥280,000
CZ-612DGY	¥290,000
CZ-605D	¥280,000

CZ-612CBK(本体)
プラス(ディスプレイ)組合せ

CZ-606D	¥330,000
CZ-605DBK	¥360,000
CZ-613DBK	¥370,000
CZ-602DBK	¥345,000

CZ-604C(本体)
プラス(ディスプレイ)組合せ

CZ-604D	特価
CZ-611DGY	¥310,000
CZ-606D	特価
CZ-605D	特価

CZ-653C(本体)
プラス(ディスプレイ)組合せ

CZ-602DBK	特価
CZ-606D	特価
CZ-612DGY	¥290,000
CZ-605D	特価

CZ-602C(本体)
プラス(ディスプレイ)組合せ

CZ-606D	¥270,000
CZ-613DGY	¥310,000
CZ-606DGY	¥300,000
CZ-611DGY	¥285,000

CZ-613CBK(本体)
プラス(ディスプレイ)組合せ

CZ-604D	¥410,000
CZ-605D	¥430,000
CZ-613D	¥440,000
CZ-21HD	¥450,000

CZ-623CTN(本体)
プラス(ディスプレイ)組合せ

CZ-611DGY	¥445,000
CZ-612DGY	¥460,000
CZ-613DTN	特価
CZ-21HD	特価

パソコンゲームソフト(X1, X1t対応)

品名	機種	特価
トンネルズ&トロールズ	X1 X1t	[illegible]
ロードウォー2000	X1	[illegible]
イースII	X1	[illegible]
ソーサリアン	X1t	[illegible]
ソーサリアン(ユーティリティー)	X1	[illegible]
ソーサリアン No.1	X1	[illegible]
ソーサリアン No.2	X1	[illegible]
ソーサリアン No.3	X1t	[illegible]
[illegible]	X1t	[illegible]
ラスト ハルマゲドン	X1t/Z	[illegible]
ランペルール	X1t	[illegible]
ザナドゥ	X1/X1t	[illegible]
[illegible]	X1t	[illegible]
大航海時代	X1t	[illegible]
アークス	X1t	[illegible]
[illegible]	X1t	[illegible]
エグザイル	X1t	[illegible]
上海	X1/X1t	[illegible]
マスターオブモンスターズ	X1t	[illegible]
ウイザードリー	X1/t	[illegible]
ワープルぷぁーしゅんず	X1t	[illegible]
サイオブレイド	X1t	[illegible]
[illegible]	X1t	[illegible]
[illegible]	X1t	[illegible]
水滸伝	X1/t	[illegible]
[illegible]	X1/t	[illegible]
マシン語ゲーム	X1/t	[illegible]
構造化BASICのすすめ	X1/t	[illegible]
マクロアセンブラー MACRO-80	X1/t	[illegible]
[illegible]	X1/t	[illegible]
ファンタジー III	X1/t	[illegible]
[illegible]	X1	[illegible]
[illegible]	X1t	[illegible]
ディスクモーター	X1/t	[illegible]
JETターボターミナル	X1t	[illegible]
三國志	X1/t	[illegible]
運動特定	X1	[illegible]
[illegible]	X1/X1t	[illegible]
CZ II LFC	X1/t	[illegible]
CZ II LF FORTRAN	X1/t	[illegible]

アイビット推奨ディスプレイ






特価¥76,800
特価¥54,000
特価¥59,800
特価¥75,300
特価¥50,800

※シャープ製品特価(機器・プリンター等)を豊富取り揃っております

富士通 FM R80 FM NOTE →特価¥190,000



SHARP AX286N-H2 All In.Note →特価¥358,000



NEC PC-98HA NOTE →特価¥159,000



TOSHIBA J-3100GS Dyna Book →特価¥155,000



富士通FM TOWNSお買得セット

FM TOWNS モデル1基本セット / FM TOWNS モデル2基本セット / FM TOWNS モデル1[illegible]セット / FM TOWNS モデル1S基本セット / FM TOWNS モデル1S[illegible]モデル2セット / FM TOWNS モデル2Fセット

TOWNSお買い上げの方は、パソコン教室が御利用できます。初・中・上級 無料にて実施中!

〈全機種価格完全保証付〉

0426-45-3002(直通) -3001(本 部) -3003(教 室)
FAX.0426-44-6002

営業時間/10:00〜19:00 [illegible]/20:00 定休日/水曜日

SHARP SUPER XEX SHOP

アイビット電子株式会社 〒192 東京都八王子市北野町560-5

全国通販売
北海道から沖縄まで
富士銀行八王子支店 (普)1752505



クリエイト特典

- ●全商品完全保証書付(メーカー保証)
- ●全国無料配送(一部離島の方は有料になります)
- ●配達日の指定OK(日曜・祭日にかかわらずお客様のご都合にあわせて配達します)
- ●どんな商品の組合せも特価提供(ご予算・用途に応じ自由にシステムアップできます)
- ●中古パソコン高額下取り(今お使いのパソコンをわずかな差額でグレードアップ)
- ●お支払い方法豊富(低金利の均等払い、ボーナス一括払いもご利用ください)

営業時間(定休日▶渋谷店:日曜・祭日/横浜店:水曜)
AM10:00〜PM7:00

当社はX68000の販売認定店です。どんなことでも安心してご相談ください。

冬のボーナスフェア実施中! 即売・即納

X68000 NEW PRO II

●CZ-653C 本体	¥285,000
●CZ-603D(カラーディスプレイ)	¥84,800
●お好きなゲームソフト1本	¥7,800
■定価合計	¥377,600

クリエイト特価

均等払い	¥7,380×48回	¥8,990×36回	¥14,370×24回
ボーナス	なし	なし	なし

X68000 EXPERT セット

●CZ-602C GY	¥355,000
●CZ-603D GY	¥84,800
■定価合計	¥440,800▶大特価¥279,000

大特価¥279,000

均等払い	¥12,860×24回	¥8,870×36回	¥7,050×48回
ボーナス	なし	なし	なし

X68000 NEW EXPERT II

●CZ-603C 本体	¥338,000
●CZ-613D(カラーディスプレイテレビ)	¥135,000
●CZ-8NJ2	¥23,800
●お好きなゲームソフト1本	¥9,800
■定価合計	¥506,600

クリエイト特価

均等払い	¥8,870×48回	¥12,040×36回	¥18,860×24回
ボーナス	なし	なし	なし

X68000 SUPER HD

●CZ-623C-TN 本体+マウス	¥498,000
●CZ-613D-TN(カラーディスプレイ)	¥135,000
●CZ-8BP1	¥79,800
■定価合計	¥712,800

クリエイト特価

均等払い	¥7,220×48回	¥10,100×36回	¥13,450×24回
ボーナス	¥42,000	¥50,000	¥40,000

※本広告に掲載の全商品の価格について消費税は含まれておりません。

X68000 NEW EXPERT II

ミュージシャンセット。これもTMネットワークたよ〜!

●CZ-603C	¥338,000
●CZ-606D	¥115,000
●MU1.6(MIDIボード&ソフト)	¥39,800
●CM32L	¥69,000
●グラナダ	¥8,800
●JOYカード	¥1,800
■定価合計	¥572,400▶超特価¥456,000

X68000 NEW PRO II

ゲーマーズセット。遊んで暮らせるSET!

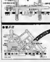
●PRO II CZ653C	¥285,000
●8.31CRT CZ603D	¥84,800
●グラナダ	¥8,800
●YS	¥8,700
●ポピュラス	¥8,800
●スーパーハングオン	¥8,800
●エーノャックス	¥8,870
●サーク	¥8,900
●アールタイプ	¥7,800
●アナログJOYSTIC XE 1AP	¥13,000
■定価合計	¥445,100▶超特価¥363,000

X68000シリーズ用 周辺機器・ソフト オール超特価!!

型番	品名	定価	ソフト名	品名	定価
CZ-6VT1	カラーイメージユニット	¥69,800	MUSIC PRO	MIDI用	¥28,800
CZ-6HS1	カラーイメージスキャナ	¥188,000	MUSIC PRO-68K	マウスを使った作曲ワープロ	¥18,800
CZ-6BE1A	1MB増設RAMボード	¥38,800	SOUND PRO-68K	サウンドエディタ	¥15,800
CZ-6BE2	2MB増設RAMボード	¥79,800	Sampling PRO-68K	AD PCMサンプルエディタ	¥17,800
CZ-6BE4	4MB増設RAMボード	¥128,000	Musician PRO-68K V2	MIDIマルチレコーディング	¥28,800
CZ-6MM3	マウス・トラックボール	¥9,800	OS-9 X68000	マルチタスクオペレーティングシステム	¥39,800
BF-68PRO	漏洩電波RTフィルター	¥19,800	PRO-68K	[illegible]	¥19,800
CZ-6BP1	有償装置プロセッサボード	¥79,800	PRO-68K	[illegible]	¥24,800
CZ-8NT1	トラックボール	¥19,800	C compiler PRO-68K	C言語	¥39,800
CZ-6BM1	MIDIボード	¥26,800	Human 68K Ver2.0	[illegible]	¥9,800
CZ-8NJ2	アナログスティック	¥23,800	PIO-6BE1-A	増設RAM	¥26,800
CZ-6TU	テレビコンチューナ	¥33,100	PIO-6BE2-2M	2MRAM	¥50,800
SX-68M	[illegible]	¥9,800	PIO-6BE4-4M	4MRAM	¥88,000
XE 1AP	アナログ	¥3,800	MU1 II	MIDI [illegible]	¥38,800

▲上記以外ビジネスソフト・最新ゲームソフト豊富に在庫あります。※送料はご注文の際お問合せください。 ●超特価遂売中!

★この欄以外の問合せ・お支払い方法もご自由にできます。
★X1シリーズ用、X68000シリーズ用各社ハードディスク/プリンタ等の周辺機器を大特価にて販売しております。
電話してお問合せください。



パソコン専門ショップ

総合お問合せ先☎03-486-6541代

ソフトクリエイト 渋谷/横浜

●渋谷店☎03-486-6541(代)
●横浜店☎045-314-4777(代) 〒221 横浜市神奈川区鶴屋町2-12-5 第1機能ビル



このキーボードは一味違う!!

あなたのX68000のキーボードをチューンナップします。

ステージI‥合計94個のキースイッチをクリック感抜群の物と交換!!

ステージII…ステージI+キーボードの101箇所に誤入力防止処理を施します。

ご注意

■LED付のキー7個、BREAK COPYキー、F1〜F11キーは、治機能上変更出来ません。
その他の入力に必要なキーを変備します。
●X68K PRO PRO IIには対応していません。

メニュー

ステージI‥¥19,800
ステージII‥¥29,800

■当社からの発送代金は全てサービスです
■消費税はいただいておりません。

通信販売のみ

ご注文は 住所・氏名・年齢・TEL・御支払方法
そして、ステージIかステージIIかを選んで、
TEL・FAX・はがき等でお申し込み下さい。
御支払方法 1 現金書留・郵便為替
2 郵便振替 横浜4-31963
3 銀行振込 協和銀行 狛江支店
当座 009867
入金確認したい梱包用の箱をお送りしますので、
あなたのキーボードを入れて御返送下さい。
当社に届きしだいすぐに作業にかかり、約一週間でお手元にお届け致します。

CYBER Corp.

株式会社 サイバー 〒227 横浜市緑区青葉台町801-32
お問い合わせは お気軽に TEL.045(982)1447 FAX 045(982)1457

▶最新刊◀



Oh!X の好評連載を単行本化!

X68000 マシン語プログラミング 〈入門編〉

村田敏幸 著
定価2,800円(税込)

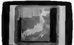
プログラミングの力は実際にプログラムすることの中から培われるという視点から、豊富な実例を示しながらマシン語プログラムのおもしろさを解説。

お近くの書店でお求め下さい。
ソフトバンク出版事業部 SOFTBANK

ACCESS

X1 エミュレータ

好評発売中
定価¥9,800



X1エミュレータはX68000上でX1シリーズのアプリケーションを実行するためのソフトエミュレータです。X1のアプリケーションを完全にソフトウェアのみでエミュレートしているため、X1上での実行速度と比較して、平均3〜5倍程度おそくなりますが、X68000のマシン上に実現した仮想X1マシンを楽しめます。また、X1とX68000の相互間でファイルを転送するためのユーティリティと専用ケーブルが付属しますので、X1上で作り上げたソフトの資産をX68000上に移行することも簡単にできます。

X1 エミュレータの機能

- X1エミュレータはX1に相当する機能をエミュレート。この仮想コンピュータには最大4つのドライブが仮想的に接続。
- X1エミュレータからみたドライブはHuman68kのドライブ上にあるファイルで仮想的に実現。このファイルはX1用の5"2Dディスクのイメージをファイル転送ユーティリティで丸ごと転送したもの。
- X1エミュレータで仮想的に実現したX1は仮想ドライブから起動。このため仮想ドライブ用ファイルには、X1を立ち上げるために必要なHuBASICやCP/Mなどのシステムプログラムが必要。
- X1エミュレータでは、X1の持つVRAMを含むメモリイメージやZ80CPUを仮想的にソフトウェアで実現。

ファイル転送ユーティリティ

ディスク転送

X1ディスク⇔X68000 Human68k(5"2Dディスクイメージファイル)

- X1エミュレータではHuman68k上のディスクイメージファイルを仮想ドライブとして使用。

ファイル転送

X1 BASIC CP/M⇔X68000 Human68k

- X1で作ったプログラム&データをX68000上で使用。

*付属の専用ケーブルをX1とX68000に接続してファイルを転送します。





X1 エミュレータ Q&A

Q. ファイル転送のために別途RS-232Cケーブルを買わないといけないのですか?
A. 専用のケーブルが付属しますのでその必要はありません。

Q. X1BASICのプログラムをX68000上のX-BASICで使えますか?
A. 通常のセーブではコードが違うので使用できませんが、アスキーセーブしたファイルであればX-BASIC上でそのままロード可能です。

Q. TurboBASICで作成した住所録などの漢字を含んだデータがあるのですがX68000上にファイル転送できますか?
A. X1TurboもX68000も漢字はシフトJISコードなのでファイルの転送は可能です。ただし、漢字ROMを必要とするものはサポートしていません。

Q. Turbo用のソフトは動きますか?
A. X1用のみでTurbo専用のソフトは動きません。

Q. ゲームは動きますか?
A. 純粋にBASICでかかれたものは動きますが、プロテクトがかかったものや直接ハードをアクセスするような市販のゲームは動きません。

*タイミング等ハードウェアに依存するようなソフトは、原理上実行できない、もしくは正常に動作しない場合がありますのでご注意ください。
*一部サポートしていない機能があります。

X1エミュレータ通信販売 購入希望として住所・氏名・電話番号をお知らせください。注文書をお送り致します。

*この価格表示には消費税は含まれておりません。
*CP/Mはデジタルリサーチ社の商標です。
文中のソフトウェアは各社の商標です。
*製品の仕様、価格は予告なく変更する場合もございますのであらかじめご了承ください。

有限会社アクセス 〒101 東京都千代田区神田神保町1-54 神保町協和ビル7F ☎03(233)0200代 FAX 03(291)7018

パソコン/ワープロ通信ネットワークサービス

J&P HOT LINE

SIG探訪PART7

ディオニュソス(ジャンプコード SAKE)

こんな人、ぜひおいで〜

若いネットワーカーが多いなか、ポツンと話題に乗り切れない私たち(オヘ S)のようなおじさんネットワーカー(女性も含む)にも専用ボード「おしんぼ"赤ちょうちん」を設けています。
「もちろん、未成年者には厳禁"未成年"も用意しています。」 by COPA2



90 8 3「かみなり亭」にて
大規模なオフミは、年間5〜6回。主に当SIG御用達「かみなり亭」(大阪谷町6丁目)での宴です。あまりの楽しさに、はるばる東京からの参加や、ボードに出ないでもミートだけは来られる方もチラホラ。

ほのぼのとした日だまりのような ぬくもりが心地よい。

一人ひとりがどんなにあがいても、何事もなかったかのように流れる世の中。やっぱり、自分を中心に波紋が広がってくれるような世界があってもいい——。**メインテーマがないSIGだけど、ボードに投げた小石(MSG)の波紋だけはきっちり広がってくれます。** 酒ボードもさることながら雑ボートが盛況で、ここを練習部屋にしている美人達の魅力にひかれ、アクセスするたびに未読MSGの山・山・山。**毎週土曜23時の定例OLTも大盛況**で、毎回システム終了時刻まで賑わっています。また、面倒見の良さも抜群で、**パソ通関の技術的なことは@ubopのLOT氏か、その他の人生相談事は@ubopの化石人類氏が、明朗闊達親身(?)な回答**をさせていただいております。

その他楽しいメニューがまだまだいっぱい!

- ★J&Pならではのパソコン・関連製品の会員割引もあるONLINE SHOPPING。
- ★J&Pだから強い!!パソコン情報をはじめとする役に立つDATA BASE。
- ★みんなでおしゃべりオンライントーク(CHAT機能)。
- ★地域別、テーマ別ボードで充実のBBS(電子掲示板)。
- ★ビジュアルデータもばっちり送受信できるX MODEM。

J&P HOT LINEへのご入会はスタータキットで。

買ったその日から1週間無料でアクセスできます。

お求め店、下記の本部へ。又は現金書留にて ¥3,000+¥90(消費税3%)=¥3,090を事務局宛でお送り下さい。すぐにスタータキットをお送りします。

お問い合わせは 〒556 大阪市浪速区日本橋 上新電機株式会社 J&P HOT LINE事務局 TEL.(06)632-2521

スタータキットのお求めはJ&P各店でどうぞ。

[illegible]

Oh!X 1月号 1991年1月1日発行（毎月1回1日発行） ソフトバンク株式会社発行 Printed in Japan 特別定価780円